LES RESEAUX DE NEURONES ARTIFICIELS

INTRODUCTION AU CONNEXIONNISME

COURS, EXERCICES ET TRAVAUX PRATIQUES

Claude TOUZET

Juillet 1992

Remerciements

De nombreuses personnes ont contribués scientifiquement, intellectuellement ou techniquement à la rédaction de cet ouvrage. Dans tous les cas, leur amitié m'honore et je tiens à leur exprimer ici ma gratitude, en particulier, le professeur Norbert Giambiasi, Directeur du LERI (Laboratoire d'Etudes et Recherche à Nîmes), l'EERIE (Ecole pour les Etudes et la Recherche en Informatique et Electronique à Nîmes) où ce cours a été proposé aux élèves de dernière année dès 1990, Mr. Jean-Claude Rault, éditeur (EC2 à Paris), toute l'équipe Neuromimétique du LERI dont nous retrouverons en partie les travaux et certains membres, éminents et sympathiques, de la communauté réseaux de neurones artificiels tels que Jeanny Herault (INPG, Grenoble), Christian Jutten (LTIRF, Grenoble), Jean-Claude Gilhodes (Lab. de Neurobiologie Humaine, Marseille).

Le LERI est, et restera, pour moi un cadre de travail stimulant et chaleureux. Je tiens à exprimer ici mon amitié à ses membres et à ceux qui ont su devenir mes amis comme Mourad Oussalah, Martine Magnan, Jean-François Santucci, Anelise Courbis, Norbert Giambiasi, Claudia Frydmann, Marc Boumedine, François Blayo, Anne Marion, Yves Coiton, Anne Guérin, Kamel Djafari, ...

D'autres ont su m'encourager, sans faillir, par leur enthousiame pour ce projet ; je dédie donc cet ouvrage à Catherine, Martine et Michel, Bernard, mes parents et grands-parents.

Introduction

L'informatique est la science du traitement automatique de l'information. Son développement est souvent confondu avec celui des machines de traitement : les ordinateurs. Depuis les débuts (ENIAC 1946) jusqu'à aujourd'hui, les ordinateurs sont devenus de plus en plus puissants.

Cependant, cette augmentation de puissance ne permet pas de toujours résoudre les problèmes d'une application informatique dans un domaine particulier. L'idée s'est donc installée que ce n'était peut être pas tant le matériel que le logiciel qui pêchait par manque de puissance. La construction de logiciels s'appuie sur plusieurs approches. Deux parmi les plus utilisées sont l'approche algorithmique et l'approche basée sur la connaissance.

Une approche algorithmique nécessite l'écriture (avant la transcription dans un quelconque langage de programmation) du processus à suivre pour résoudre le problème. Lorsque le problème est complexe, ce peut être une étape coûteuse ou impossible. D'autre part, les ordinateurs sont des machines complètement logiques (et même binaires) qui suivent à la lettre chacune des instructions du programme. C'est un avantage lorsque tous les cas ont été prévus à l'avance par l'algorithmicien. Ce n'est hélas pas toujours possible. Dans ce cas, dixit l'informaticien : "c'est une faute de la machine". Rien de plus faux ! Ainsi les systèmes informatiques embarqués (à bord des avions, de la navette spatiale, etc) tentent de pallier à ce manque (prévisible) de clairvoyance de l'algorithmicien en triplant les logiciels, chacun étant développés indépendamment par une équipe différente, dans des langages différents. Les risques de laisser l'ordinateur aux prises avec une situation imprévue, où son comportement ne serait pas adapté, sont ainsi considérablement réduits. Rappellons-nous le haro lancé sur les programmes boursiers lors de la chute de la bourse en 1987.

La seconde approche possible est celle de l'intelligence artificielle (appelée IA par commodité), avec pour applications les plus connues les systèmes experts. Ici, la résolution du problème est confiée à un ensemble de règles données par l'expert humain du domaine. Il n'en demeure pas moins que toutes les règles doivent avoir été exprimées préalablement au traitement, et que le programme demeure binaire dans son exécution. Les cas qui n'ont pas été prévus par l'expert ne seront pas correctement traités. L'introduction de la logique floue ne change pas la nature des limitations d'emploi du programme : l'exécution reste totalement déterministe. En fait, l'approche basée sur la connaissances se limite à des domaines d'application où la modélisation de la connaissance, par exemple sous forme de règles, est possible. Ces domaines sont souvent ceux des sciences dites "exactes" comme l'électronique, la mécanique, la physique, etc, par opposition aux sciences dites "humaines" comme la médecine, la psychologie, la philosophie, etc, où la connaissance est plus empirique. L'IA se révèle donc être principalement un moyen commode de stocker de la connaissance sous forme explicite.

Ces deux approches ne suffisent pas à répondre à tous les problèmes existants. Citons les domaines de la reconnaissance de formes (images ou signaux), du diagnostic, du contrôle moteur, de la traduction automatique, de la compréhension du langage, depuis longtemps explorés à l'aide des approches algorithmiques et à base de connaissances, qui n'ont pas rencontré le succès escompté. Pourtant, des êtres vivants relativement simples sont capables de réaliser certaines de ces opérations apparemment sans difficulté. Il suffit pour s'en rendre compte de lever les yeux, suivre le vol de la mouche et essayer de la capturer. Que dire alors du déplacement au sonar de la chauve souris, etc.

Une troisième approche au traitement automatique de l'information semble donc s'offrir à nous, où l'on cherche à s'inspirer du traitement de l'information effectué par le cerveau. L'hypothèse principale, à la base de l'essort des réseaux de neurones artificiels, est que le comportement intelligent est sous-tendu par un ensemble de mécanismes mentaux. Ces mécanismes étant basés sur des processus neurophysiologiques, nous supposons donc que la structure du système nerveux central est à la base du développement d'un comportement intelligent. Remarquons que cette hypothèse n'a pas toujours eu cours. Ainsi, depuis l'antiquité, le siège des émotions a lentement migré depuis les pieds, vers l'estomac (qui se noue face au danger), puis le coeur (qui s'accélère lors des passions) pour finir dans la boite crânienne.

La figure 1 reprend l'hypothèse proposée par de nombreux biologistes : pour recréer le comportement intelligent du cerveau, il faut s'appuyer sur son architecture, en fait, tenter de l'imiter.

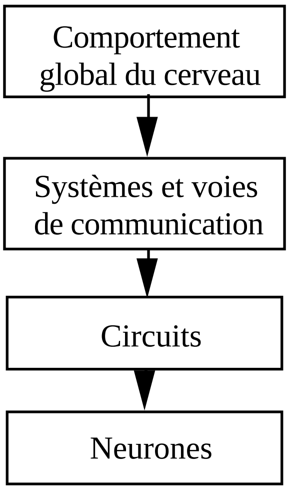


Figure 1. Hypothèse biologique de génération d'un comportement intelligent

Ce dernier paragraphe nous fournit déjà le plan de cet ouvrage. Dans une première partie, nous examinons quelques notions biologiques relatives au cerveau, et à ses constituants les neurones et leurs synapses. L'organisation en réseaux des neurones permet d'illustrer les notions d'apprentissage et de mémorisation (modification des connexions). Ces données nous

sont nécessaires pour aborder le second chapitre qui montre le passage des modèles de réseaux neuronaux biologiques à des modèles mathématiques : les réseaux de neurones artificiels. Nous établissons un tableau des correspondances biologique/artificiel, avec notamment des modèles de neurones et de synapses et quelques topologies pour l'organisation en réseaux. Au travers d'un exemple simple, nous décrivons le fonctionnement des réseaux de neurones artificiels et leurs propriétés d'apprentissage à partir d'exemples, de résistance au bruit, d'adaptabilité et de tolérance au pannes. Il existe de nombreux modèles de réseaux de neurones artificiels, nous en présentons successivement quelques uns choisis principalement selon des critères de nature pédagogique. Le Perceptron est historiquement le premier modèle, son fonctionnement est particulièrement intéressant pour le suite de notre étude. De fait, il demeure un modèle de base, de même que les cartes auto-organisatrices plus vraisemblables d'un point de vue biologique. Ces deux modèles nous permettent d'introduire les concepts d'apprentissage supervisé et non supervisé. Des modèles plus élaborés sont étudiés par la suite tant au niveau de leur architectures, des techniques d'apprentissage que des performances. Ce sont les mémoires associatives, le réseau ART et une version plus complexe et surtout plus efficace du Perceptron : le Perceptron multicouche.

Connaître les modèles est d’un profond intérêt, mais pour l’ingénieur le développement d’une application basée sur les réseaux de neurones artificiels peut sembler plus important. Nous consacrons un chapitre à la reconnaissance de caractères manuscrits réalisée par un Perceptron multicouche. Ecrire des programmes de simulations pour quelques modèles de réseaux est du plus haut intérêt pédagogique. Cependant le développeur dispose aujourd’hui d’environnements de développement pratiques et puissants dont nous passons en revue les principales caractéristiques.

S'agissant d'un ouvrage de vulgarisation à l'usage des étudiants de tous les ages, nous avons tenu, en nous basant sur notre expérience d'enseignement, à proposer les outils pédagogiques que sont les exercices et les travaux pratiques. Il s'agit bien entendu d'aider le lecteur à vérifier sa compréhension des concepts, des modèles et de le familiariser à la manipulation des algorithmes. Nous espérons que vous vous impliquerez dans ce "surplus" de travail proposé. Toutes les réponses se trouvent évidemment dans ce livre.

1 Les réseaux de neurones artificiels

1 Définition

Aujourd’hui de nombreux termes sont utilisés dans la littérature pour désigner le domaine des réseaux de neurones artificiels, comme connexionnisme ou neuromimétique. Pour notre part, il nous semble qu'il faut associer à chacun de ces noms une sémantique précise. Ainsi, les réseaux de neurones artificiels ne désignent que les modèles manipulés ; ce n’est ni un domaine de recherche, ni une discipline scientifique. Connexionnisme et neuromimétique sont tous deux des domaines de recherche à part entière, qui manipulent chacun des modèles de réseaux de neurones artificiels, mais avec des objectifs différents. L’objectif poursuivi par les ingénieurs et chercheurs connexionnistes est d’améliorer les capacités de l’informatique en utilisant des modèles aux composants fortement connectés. Pour leur part, les neuromiméticiens manipulent des modèles de réseaux de neurones artificiels dans l'unique but de vérifier leurs théories biologiques du fonctionnement du système nerveux central. Notons qu’en France, dès 1982, des réunions de ces deux communautés ont été organisées, ce sont les Journées Neurosciences et Sciences de l’Ingénieur (cf. chp. Informations pratiques).

Le titre même de cet ouvrage ne laisse aucun doute, nous nous plaçons du point de vue de l’ingénieur à la recherche d’une connaissance connexionniste. Ceci nous oblige cependant à aborder au chapitre suivant des notions de neurosciences utiles à notre projet.

Définition :

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

2 Historique

- 1890 : W. James, célèbre psychologue américain introduit le concept de mémoire associative, et propose ce qui deviendra une loi de fonctionnement pour l’apprentissage sur les réseaux de neurones connue plus tard sous le nom de loi de Hebb.
- 1943 : J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ceux sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).
- 1949 : D. Hebb, physiologiste américain explique le conditionnement chez l’animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement de type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraîne chez cet animal la sécrétion de salive à

cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux.

2 Les premiers succès

- 1957 : F. Rosenblatt développe le modèle du Perceptron. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance de formes. Notons qu'à cet époque les moyens à sa disposition sont limités et c'est une prouesse technologique que de réussir à faire fonctionner correctement cette machine plus de quelques minutes.
- 1960 : B. Widrow, un automaticien, développe le modèle Adaline (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétropropagation de gradient très utilisé aujourd'hui avec les Perceptrons multicouches. Les réseaux de type Adaline restent utilisés de nos jours pour certaines applications particulières. B. Widrow a créé dès cette époque une des premières firmes proposant neuro-ordinateurs et neuro-composants, la "Memistor Corporation". Il est aujourd'hui le président de l'International Neural Network Society (INNS) sur laquelle nous reviendrons au chapitre Informations pratiques.
- 1969 : M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tous modèles de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux U.S.A.), les chercheurs se tournent principalement vers l'IA et les systèmes à bases de règles.

3 L'ombre

- 1967-1982 : Toutes les recherches ne sont, bien sûr, pas interrompues. Elles se poursuivent, mais déguisées, sous le couvert de divers domaines comme : le traitement adaptatif du signal, la reconnaissance de formes, la modélisation en neurobiologie, etc. De grands noms travaillent durant cette période tels : S. Grossberg, T. Kohonen, ... dont nous reparlerons.

4 Le renouveau

- 1982 : J. J. Hopfield est un physicien reconnu à qui l'on doit le renouveau d'intérêt pour les réseaux de neurones artificiels. A cela plusieurs raisons :

Au travers d'un article court, clair et bien écrit, il présente une théorie du fonctionnement et des possibilités des réseaux de neurones. Il faut remarquer la présentation anticonformiste de son article. Alors que les auteurs s'acharnent jusqu'alors à proposer une structure et une loi d'apprentissage, puis à étudier les propriétés émergentes ; J. J. Hopfield fixe préalablement le comportement à atteindre pour son modèle et construit à partir de là, la structure et la loi

d’apprentissage correspondant au résultat escompté. Ce modèle est aujourd’hui encore très utilisé pour des problèmes d'optimisation.

D'autre part, entre les mains de ce physicien distingué, la théorie des réseaux de neurones devient respectable. Elle n’est plus l’apanage d’un certain nombre de psychologues et neurobiologistes hors du coup.

Enfin, une petite phrase, placée en commentaire dans son article initial, met en avant l’isomorphisme de son modèle avec le modèle d’Ising (modèle des verres de spins). Cette idée va drainer un flot de physiciens vers les réseaux de neurones artificiels.

Notons qu’à cette date, l’IA est l’objet d’une certaine désillusion, elle n’a pas répondue à toutes les attentes et s’est même heurtée à de sérieuses limitations. Aussi, bien que les limitations du Perceptron mise en avant par M. Minsky ne soient pas levées par le modèle d’Hopfield, les recherches sont relancées.

5 La levée des limitations

- 1983 : La Machine de Boltzmann est le premier modèle connu apte à traiter de manière satisfaisante les limitations recensées dans le cas du perceptron. Mais l’utilisation pratique s’avère difficile, la convergence de l’algorithme étant extrêmement longue (les temps de calcul sont considérables).
- 1985 : La rétropropagation de gradient apparaît. C’est un algorithme d‘apprentissage adapté aux réseaux de neurones multicouches (aussi appelés Perceptrons multicouches). Sa découverte réalisée par trois groupes de chercheurs indépendants indique que "la chose était dans l’air". Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d’entrée/sortie sur un réseau en décomposant cette fonction en une suite d’étapes linéairements séparables. De nos jours, les réseaux multicouches et la rétropropagation de gradient reste le modèle le plus étudié et le plus productif au niveau des applications. Nous lui consacrons quelques chapitres.

6 La situation actuelle (1992)

En France, elle est à l’image du congrès Neuro-Nîmes qui a pour thème les réseaux neuromimétiques et leurs applications. Créé en 1988, le chiffre de ses participants croit chaque année et reflète bien l’intérêt que le monde scientifique et industriel (50% des participants) porte au connexionnisme (fig. 1).

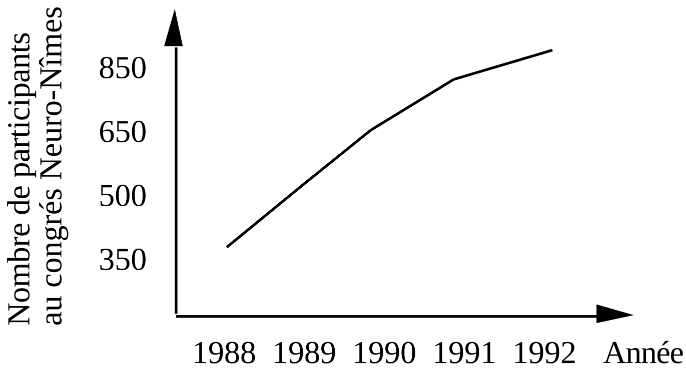


Figure 1. Illustration de l'accroissement d'intérêt pour les réseaux de neurones : évolution du nombre de participants au congrès Neuro-Nîmes

Dans le monde, et en particulier aux U.S.A., l'intérêt pour les réseaux de neurones a démarré plus tôt. Dès 1986, de 600 à 2000 visiteurs participent aux quelques grands congrès annuels.

Au niveau commercial, la figure 2 montre que plus de 200 compagnies sont aujourd'hui impliquées dans des développements d'applications connexionnistes.

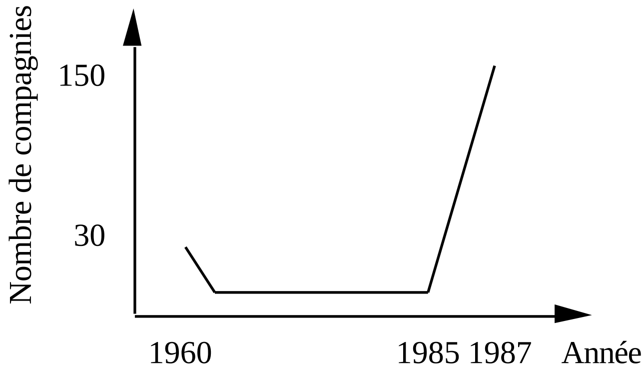


Figure 2. Evolution du nombre de compagnies proposants des produits connexionnistes (d'après DARPA 88)

Les prévisions du marché se chiffrent déjà en dizaines de millions de dollars. Il devrait dépasser les 100 millions de dollars dès 1992.

Un coup d'oeil plus détaillé aux différentes parts de marché (fig. 3) montre une évolution vers la mise en place de puces spécialisées, le développement d'applications spécifiques ou standardisées et la réduction de la partie formation.

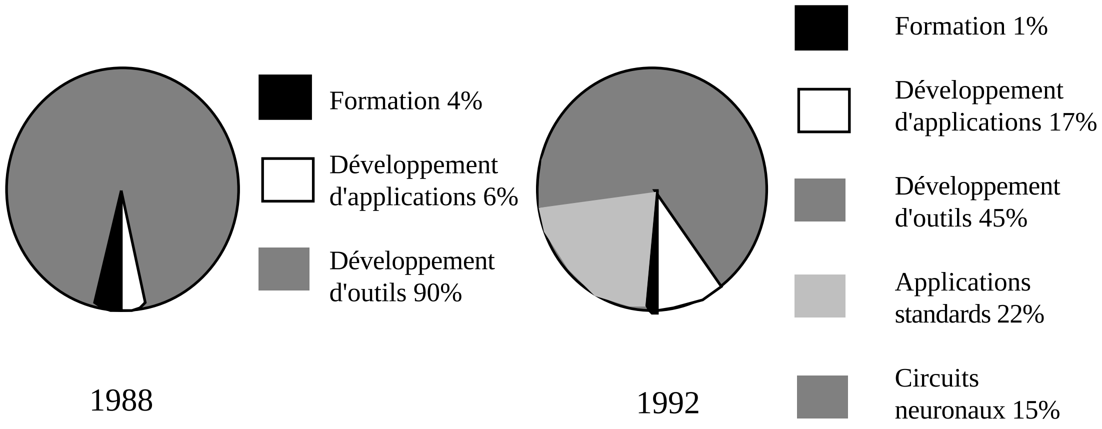


Figure 3. Evolution des différentes parts du marché connexionniste (d'après DARPA 88)

La réduction de la partie formation est le fait d’une théorie des réseaux de neurones de mieux en mieux comprise, plus facilement expliquée et appartenant de plus en plus souvent au bagage scientifique des jeunes universitaires et ingénieurs. Un enseignement spécifique réseaux de neurones artificiels a d'ailleur débuté à l’UCSD (University of California at San Diego) dès 1982. En France, universités et écoles d’ingénieurs proposent en troisième cycle de quelques heures à quelques dizaines d’heures sur ce sujet. Nous en donnons à la fin de cet ouvrage, au chapitre des informations pratiques, une liste non exhaustive.

2 Le modèle neurophysiologique

Le cerveau se compose d'environ 10^{12} neurones (mille milliards), avec 1000 à 10000 synapses (connexions) par neurone. Nous allons dans ce chapitre décrire succinctement l'élément de base du système nerveux central : le neurone. L'étape suivante nous conduit à l'étude de petits réseaux de neurones, tels ceux impliqués dans les arcs réflexes. Ceci nous amène à exposer les propriétés d'habituation, de sensibilisation et surtout à concevoir l'idée d'une modification physique des connexions entre neurones pour supporter ces phénomènes. L'étude du mécanisme de la vision chez l'animal (et l'homme) permet d'appréhender les notions de messages somato-sensoriels, de réduction d'information, d'étages de traitement et de complexification de l'information.

1 Le neurone

1.1 Structure

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angstroms (10^{-9} m) entre l'axone du neurone afférent et les dendrites (on dit *une* dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse (fig. 1).

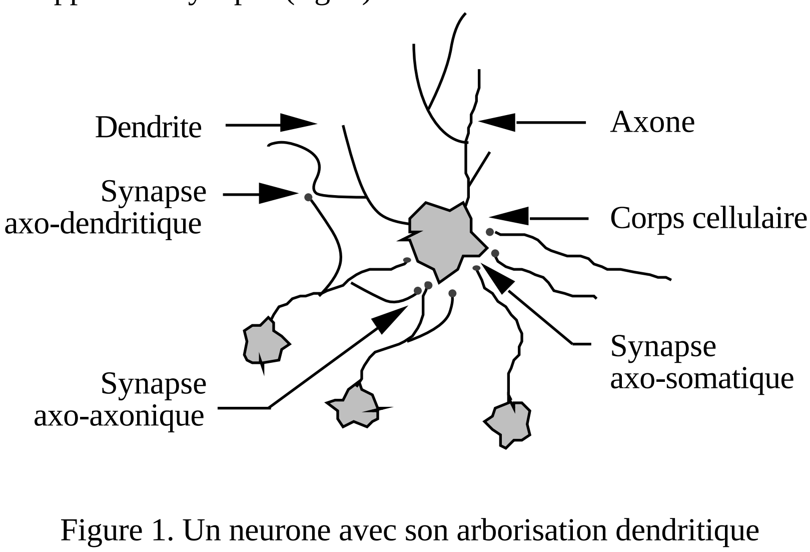


Figure 1. Un neurone avec son arborisation dendritique

Selon le type du neurone (fig. 2), la longueur de l'axone peut varier de quelques microns à 1,50 mètres pour un moto-neurone. De même les dendrites mesurent de quelques microns à

1,50 mètres pour un neurone sensoriel de la moelle épinière. Le nombre de synapses par neurone varie aussi considérablement de plusieurs centaines à une dizaine de milliers.

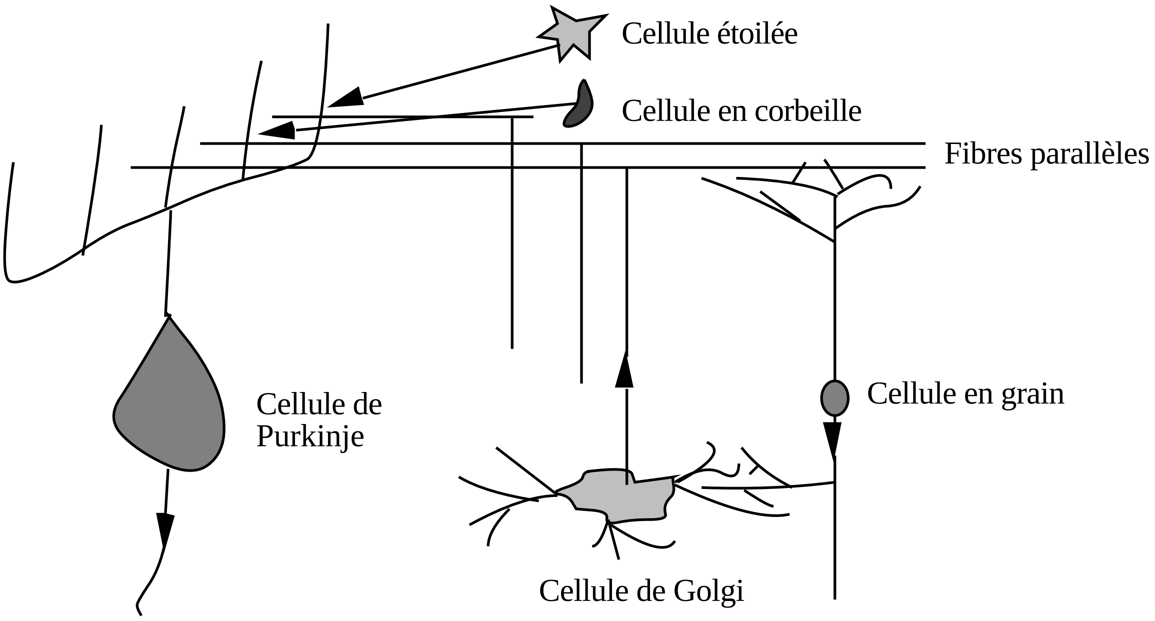


Figure 2. Description schématique des divers types structuraux de neurones présents dans le cortex cérébelleux. Les axones ont été repérés par une flêche.

1.2 Physiologie

La physiologie du neurone est liée aux propriétés de la membrane nerveuse et au métabolisme de la cellule. La différence de potentiel mesurée entre le milieu intérieur de la cellule et le milieu extérieur est de -60 mV. Pour maintenir une telle différence de potentiel, la cellule fait appel à des pompes ioniques (Na^+, K^+, ...). Cependant, une faible dépolarisation de la membrane entraine une certaine perméabilité aux ions sodiums (Na^+), dont l'effet peut être catastrophique au niveau cellulaire. En effet, à partir d'une certaine valeur seuil de dépolarisation de la membrane, il y a rupture des équilibres ioniques et création d'un potentiel d'action (aussi nommé "spike" en anglais, fig. 3).

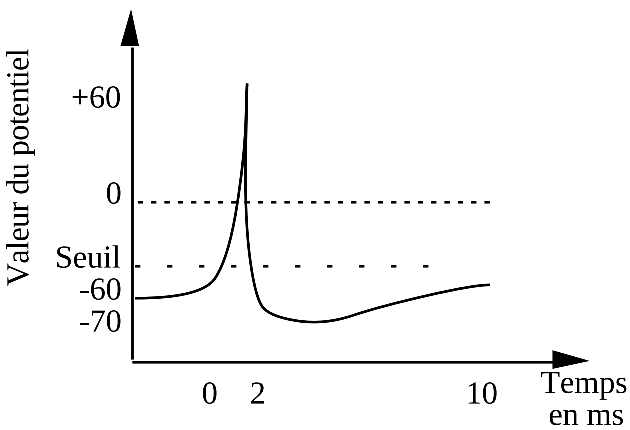


Figure 3. Un potentiel d'action

Les ions, Na^{+} en particulier, s'engouffrent en nombre dans la cellule (aidés en cela par l'ouverture des canaux Na^{+} et une différence de potentiel très attirante de -60 mV). En une miliseconde, la différence de potentiel devient égale à +60 mV (fig. 4). En fait, à partir d'une valeur de potentiel nulle, l'équilibre ionique est établi et les ions ne devraient plus pénétrer dans la cellule. Cependant, l'effet d'entrainement est tel que cette valeur d'équilibre théorique est largement dépassée. Les différents canaux ioniques se referment alors, les pompes ioniques se remettent à fonctionner, rejetant à l'extérieur de la cellule les ions en excés. Là aussi, on constate un certain effet d'entrainement : le retour à la normale passe d'abord par une phase d'hyperpolarisation. Le potentiel de repos (-60 mV) est dépassé jusqu'à atteindre (-70 mV).

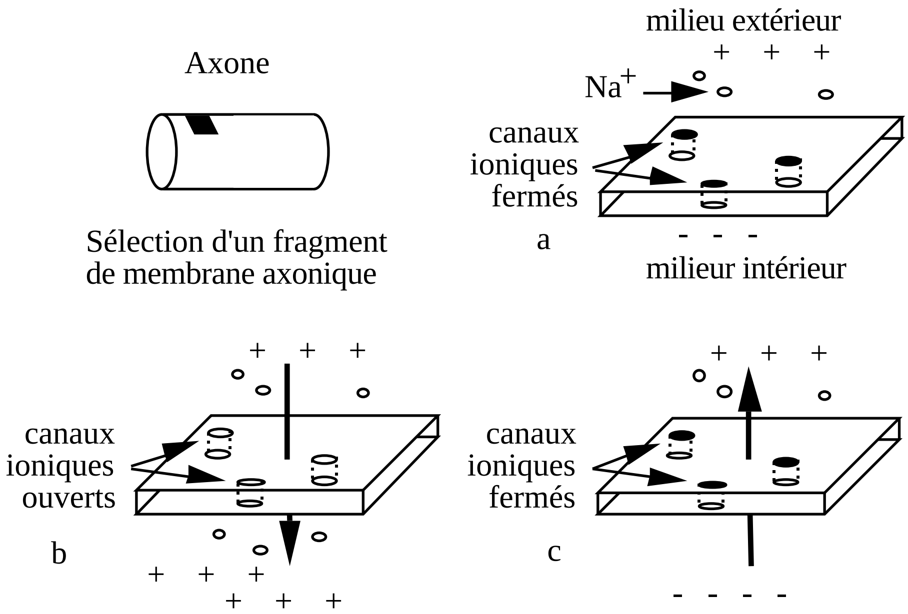


Figure 4. Passage d'un potentiel d'action au niveau de la membrane de l'axone
a) Equilibre ionique (au repos).
b) Arrivée d'un potentiel d'action (dépolarisation).
c) Après la dépolarisation : l'hyperpolarisation.

Toute cette série d'évènements cataclismiques au niveau cellulaire n'aura duré que 5 à 10 milisecondes. Durant la phase d'hyperpolarisation, le neurone est très difficilement excitable. Ce qui s'explique par le fait que la différence de potentiel par rapport à la valeur seuil (S) est plus importante que celle au repos.

1.3 Création d'un potentiel d'action

La dépolarisation initiale de la membrane axonique est créée par l'arrivée de potentiels d'action des neurones afférents sur les synapses dendritiques et somatiques. En fait, à l'arrivée

d'un potentiel d'action sur une synapse, un neuromédiateur est libéré dans l'espace synaptique. Il va ouvrir des canaux ioniques sur la membrane post-synaptique, créant ainsi une dépolarisation (aussi appelée potentiel évoqué) qui s'étend jusqu'à l'axone (fig. 5).

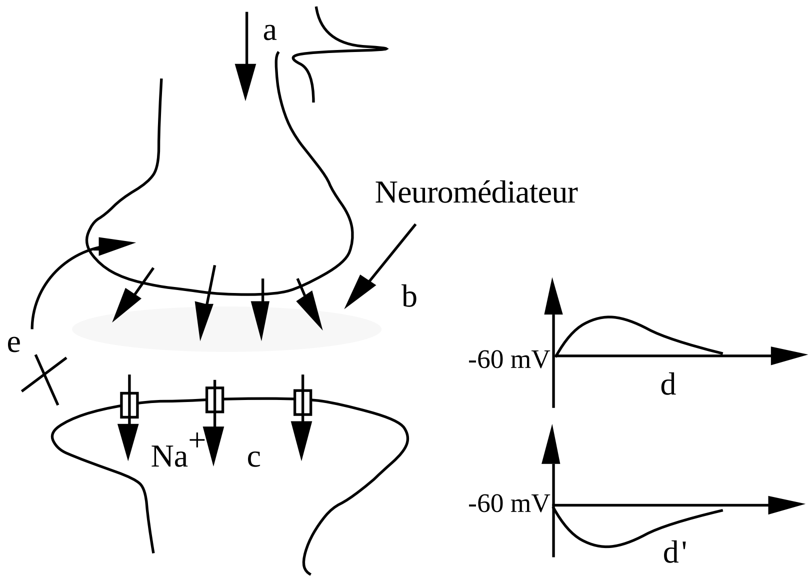


Figure 5. Fonctionnnement au niveau synaptique

a) Arrivée d'un potentiel d'action.

b) Libération du neuromédiateur dans l'espace synaptique.

c) Ouvertures des canaux ioniques dues au neuromédiateur.

d) Génération d'un potentiel évoqué excitateur.

d') Génération d'un potentiel évoqué inhibiteur. Les synapses inhibitrices empêchent la génération de potentiel d'action.

e) Fermeture des canaux, élimination ou recapture du neuromédiateur.

Les dépolarisations unitaires sont sommées dans l'espace (toutes les synapses du neurone) et dans le temps (sur une période de quelques milisecondes) et génèrent, éventuellement, un potentiel d'action sur le neurone post-synaptique. Ainsi que le montre la figure 6, la génération d'un potentiel d'action est le fruit de nombreuses dépolarisations, l'action d'une seule synapse est pratiquement sans effet.

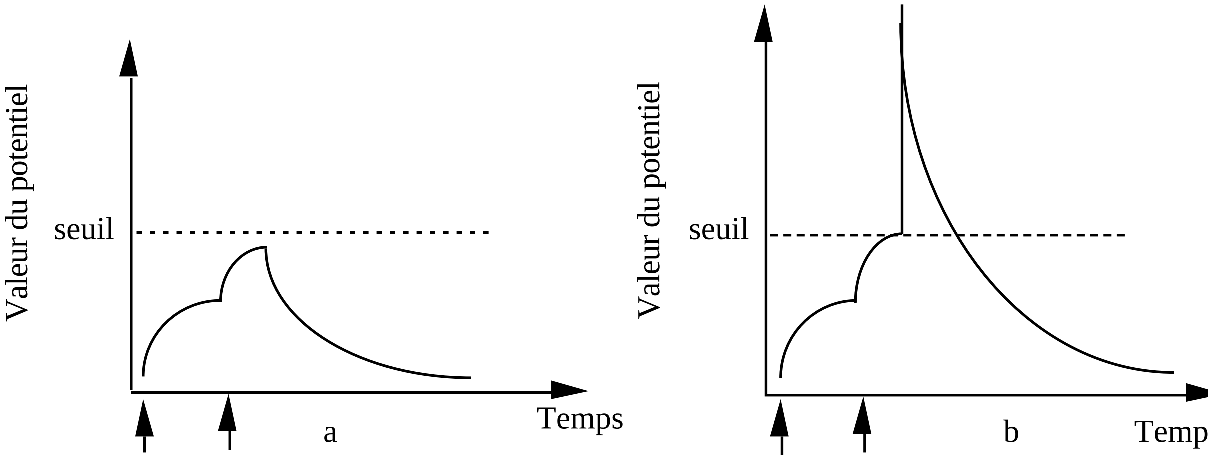


Figure 6. Sommation spatio-temporelle :
addition des potentiels évoqués à la fois dans l'espace et dans le temps.
a) 2 potentiels évoqués (repérés par les flèches) ne dépassent pas la valeur seuil.
b) 2 potentiels évoqués qui dépassant la valeur seuil génèrent un potentiel d'action.

2 Message nerveux

Le système nerveux travaille avec (entre autres) un codage en fréquence. C'est le nombre de potentiel d'action par seconde (fréquence) et les variations de fréquence (fréquence instantanée) qui code l'information. Un potentiel d'action isolé ne signifie rien. Rappelons d'autre part que tous les potentiels d'action ont la même valeur de potentiel. Par exemple (fig. 7), les messages transmis lors de mouvements du coude permettent de connaitre en fonction de la fréquence : la valeur de l'angle et en fonction des variations de fréquences : la vitesse de rotation entre deux positions.

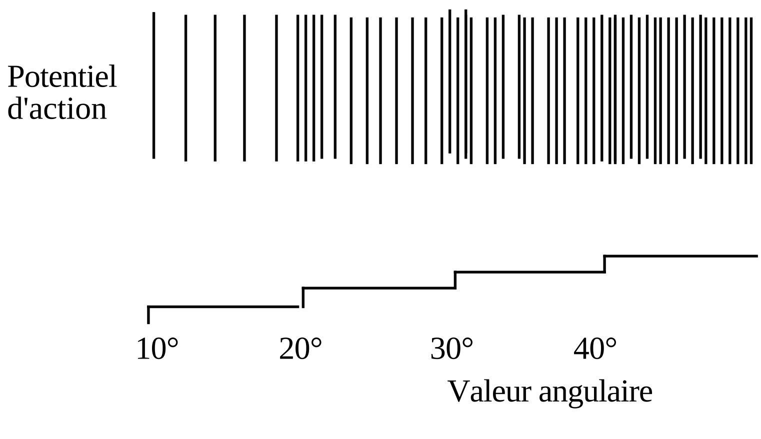


Figure 7. Exemple de codage en fréquence (mouvements d'une articulation telle que le coude).

3 Circuits neuronaux

Nous avons vu que chaque neurone est une unité autonome au sein du cerveau. Le neurone reçoit en continu des entrées. Le corps cellulaire du neurone est le centre de contrôle. C'est là

que les informations reçues sont interprétées. La réponse, unique, à ces signaux est envoyée au travers de l'axone. L'axone fait synapse sur d'autres neurones (un millier). Le signal transmis peut avoir un effet excitateur ou inhibiteur. Le traitement très simple réalisé par chaque neurone indique que l'information n'est pas stockée dans les neurones, mais est plutôt le résultat du comportement de toute la structure interconnectée. L'information est, principalement, dans l'architecture des connexions et dans la force de ces connexions.

C'est ce que nous allons vérifier avec quelques expérimentations simples réalisées sur l'aplysie (limace de mer, fig. 8). Des modifications comportementales importantes résultent de modifications simples au niveau synaptique. Les connexions renforcent ou diminuent leur efficacité (modification des forces de connexions). Dans les cas extrêmes, de nouvelles connexions apparaissent ou disparaissent (modification de l'architecture).

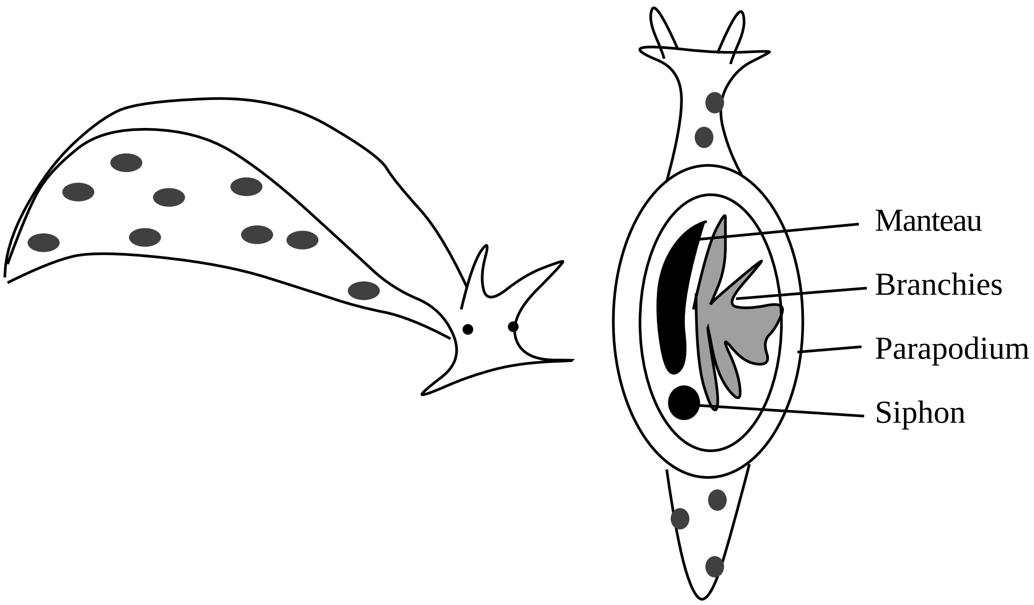


Figure 8. Aplysie ou limace de mer (abondante en méditeranée).
Au toucher du siphon ou du manteau, la contraction du siphon entraine
le retrait des branchies sous le manteau dans un reflexe de défense.

3.1 Habituation

Description de l'expérience : Le neurone sensoriel est activé par le toucher du manteau. Le neurone moteur agit alors en retractant les branchies (fig. 9). Lorsque la stimulation est répétée, la réponse de l'animal devient plus faible, jusqu'à une absence de réaction au toucher. C'est le phénomène de l'habituation (fig. 10).

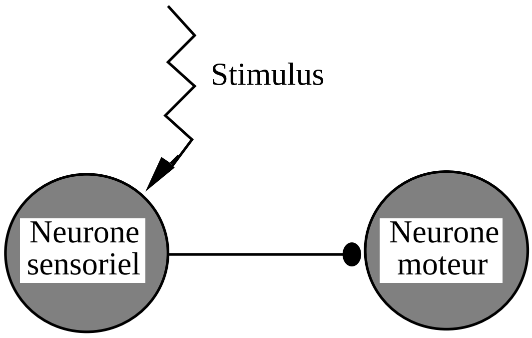


Figure 9. Circuits mis en jeu dans l'habituation

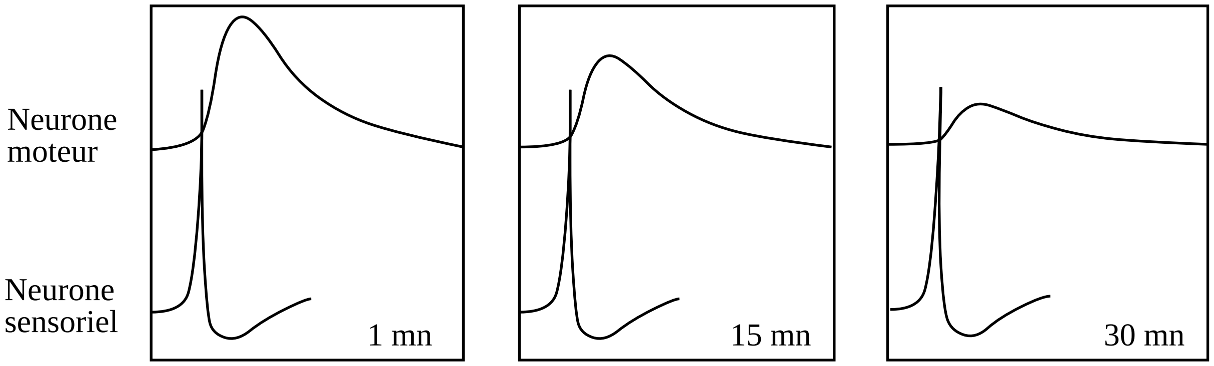


Figure 10. Habituation : lorsque la stimulation est répétée (quelques dizaines de fois), la réponse de l'animal devient de plus en plus faible, jusqu'à une absence de réaction au stimulus sensoriel. On a indiqué en bas à droite de chaque schéma le nombre de minutes aprés le début de l'expérience. A partir de 15 mn, il n'y a plus de stimulations.

3.2 Sensibilisation

Si l'on répète la même expérience en créant après chaque stimulation du manteau un courant d'eau violent qui risque d'endommager les branchies, on observe alors l'effet inverse. Le courant d'eau sert de renforcement (fig. 11) et la réponse de l'animal au stimulus initial est augmentée (fig. 12). Cet effet est appelé sensibilisation.

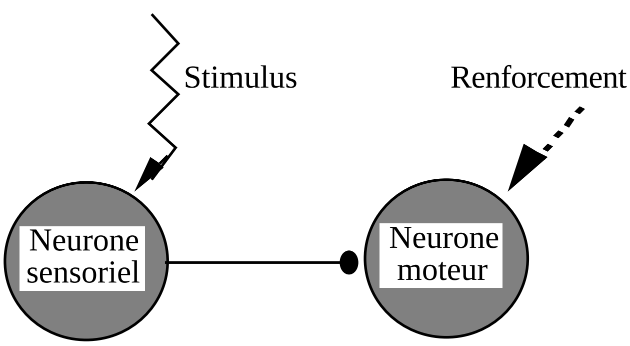


Figure 11. Circuits mis en jeu dans la sensibilisation

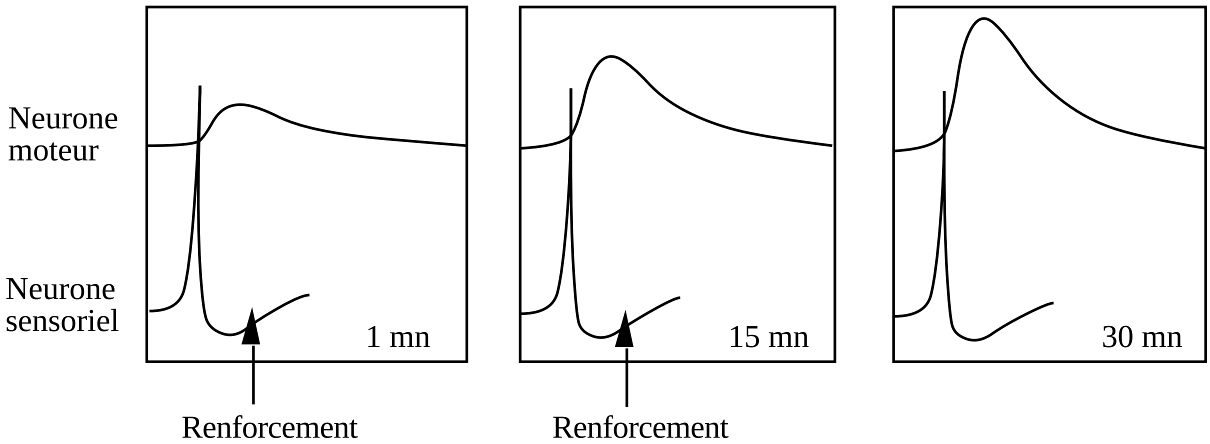


Figure 12. La sensibilisation en 3 schémas : la réponse du neurone moteur au stimulus initial est augmentée par l'action du stimulus de renforcement. Le stimulus de renforcement n'est appliqué qu'aprés la réponse motrice.

3.3 Modification synaptique

Habituation et sensibilisation au niveau neuronal traduisent la fonction d'apprentissage au niveau de l'animal dans son ensemble. Il y a adaptation de la réponse à l'environnement. L'observation des synapses mises en jeu au microscope électronique montre des modifications physiques (fig. 13).

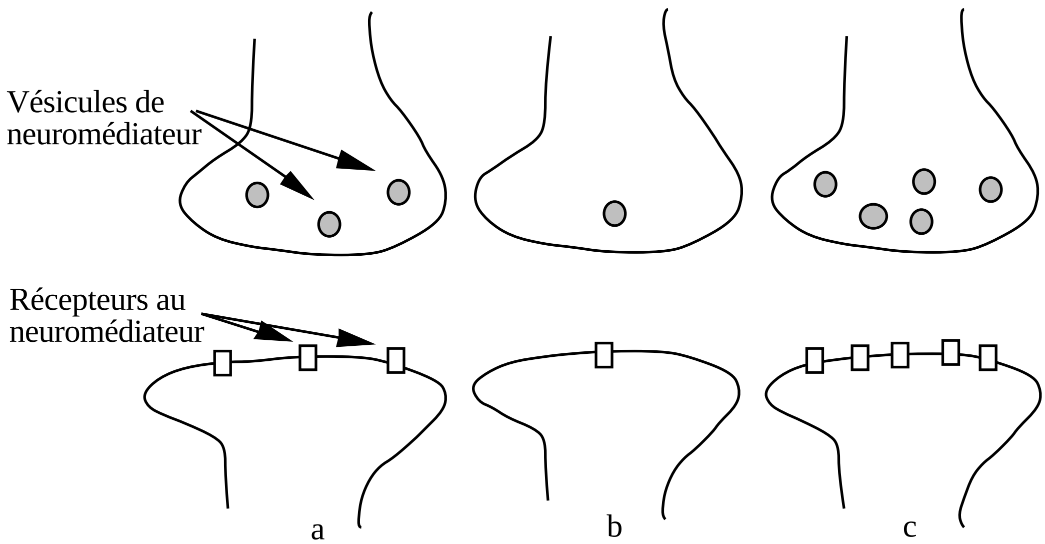


Figure 13. Modification physique de la synapse

a) Témoin.

b) Habituation : diminution du nombre de vésicules et du nombre de récepteurs.

c) Sensibilisation : augmentation du nombre de vésicules et de récepteurs.

4 La vision et les étages de traitement

Nous avons vu des mécanismes de traitement de l'information au niveau de la coopération entre deux neurones. Il existe des structures plus complexes mettant en jeu des millions de neurones, qui rangés par étages de traitement diminuent la complexité de l'information, la rendant plus signifiante. C'est le cas du système visuel, sans doute le mieux étudié aujourd'hui.

Au niveau de la rétine, plusieurs dizaines de types différents de cellules codent les informations visuelles, chacune réalisant une fonction très spécialisée. Les images sont transformées en train d'impulsions nerveuses que le nerf optique véhicule vers le cerveau. Le cerveau élabore sa perception visuelle grâce à ces signaux. Cependant, au niveau de la rétine, il y a déjà traitement de l'information. En effet, on compte environ 150 millions de batonnets et 7 millions de cônes pour seulement 1 million de fibres au niveau du nerf optique.

On connait aujourd'hui un certain nombre de circuits neuronaux de la rétine impliqués dans le traitement de l'information visuelle. Par exemple, à chaque cellule ganglionnaire correspond un champ récepteur : une zone précise du champ visuelle (disque d'un centimètre de diamètre à deux mètres de distance). Dès 1952, deux types de cellules ganglionnaires ont été répertoriés. En abscence de stimulation lumineuse (obscurité), ces cellules émettent cependant spontanément un niveau moyen de potentiels d'action. Les cellules à centre ON augmentent ce nombre d'impulsions lorsqu'un stimulus éclaire le centre du champ récepteur et deviennent silencieuses si le stimulus éclaire la périphérie du champ récepteur. Les cellules à centre OFF montrent un comportement inverse. La figure 14 montre un exemple d'architecture fonctionnelle pour une cellule ganglionnaire à centre ON. Cette oppostion de fonctionnement entre le centre et la périphérie du champ récepteur permet d'améliorer les contrastes. On a découvert depuis d'autres cellules qui codent les directions de mouvements, etc.

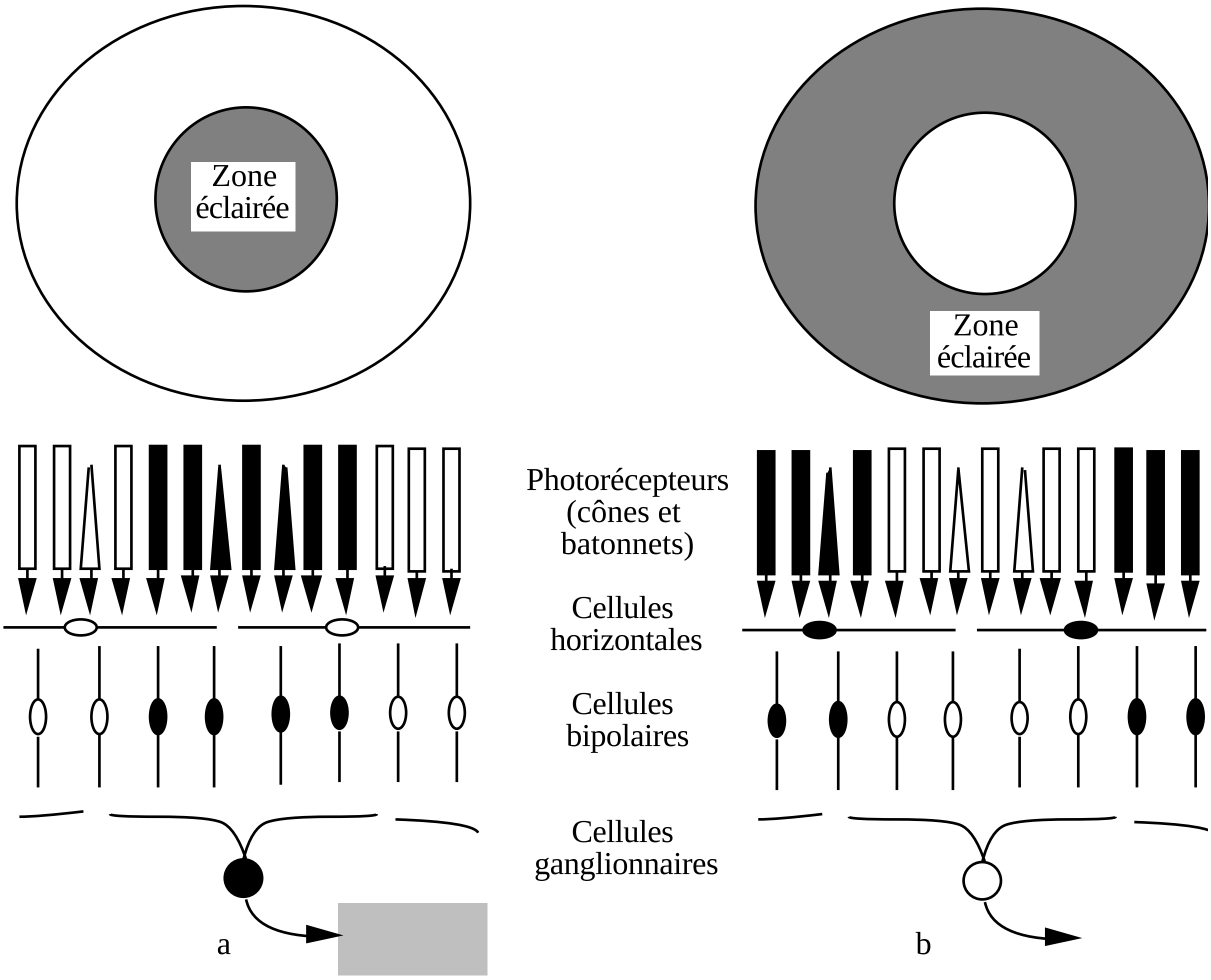


Figure 14. Exemple de traitement de l'information rétinienne par une cellule ganglionnaire à centre ON. En noir, les cellules actives. Les cellules horizontales ont une action inhibitrice sur les cellules bipolaires, elles s'opposent ainsi aux cellules photoreceptrices.
a) L'éclairage du centre du champ récepteur génère une augmentation du niveau d'activité.
b) L'éclairage de la périphérie du champ récepteur rend cette cellule silencieuse.

Au niveau du cortex visuel (arrivée du nerf optique), D. Hubel et H. Wiesel ont découvert l'existence de colonnes de dominance oculaire, spécifiquement excitées par un stimulus sous forme de barre dotée une orientation précise. La figure 15 montre une représentation schématique du cortex visuel.

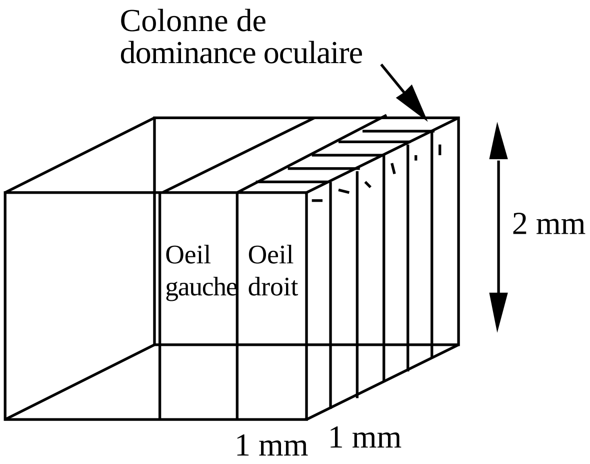


Figure 15. Représentation schématique du cortex visuel qui montre les colonnes de dominances oculaires et leur orientation privilégiée. On remarque l'alternance oeil gauche - oeil droit.

Nous avons vu une organisation topologique précise pour le traitement de l'information visuelle dont la construction semble génétique. Il existe néanmoins des possibilités d'apprentissage sur cette structure. Des expériences ont montré que l'élevage d'un chaton dans un univers composé uniquement de verticales va modifier ses perceptions jusqu'à le rendre pratiquement aveugle aux autres directions (horizontales et obliques) à l'age adulte. L'étude histologique montre que la grande majorité de ses colonnes de dominances oculaires se sont "recyclées" dans les verticales.

Quels sont les mécanismes qui permettent de modifier le comportement des structures neuronales ? D. Hebb a proposé en 1949 une règle où la force de la connexion entre deux neurones augmente si il y a corrélation d'activité (si l'activation de l'une entraine l'activation de l'autre). Cette hypothèse a depuis été complétée par J. P. Rauscheker et W. Singer qui proposent de modifier en les diminuant les forces des connexions non fonctionnelles (inutiles dans le contexte de fonctionnement actuel). Remarquons que cette loi d'apprentissage ne concerne que les synapses excitatrices, rien n'est proposé pour les synapses inhibitrices.

5 Conclusion

L’objectif pédagogique visé dans ce survol du monde biologique est la mise en exergue d'une organisation structurelle des neurones. Chaque structure est dotée d'une fonction particulière et ces structures adaptent leur comportement par des mécanismes d’apprentissage. L’apprentissage implique des modifications physiques des connexions entre neurones. L’association entre plusieurs structures neuronales, dotées chacune d’une fonction précise, permet l’émergence d’une fonction d’ordre supérieure pour l’ensemble.

3 Les modèles mathématiques

Les réseaux de neurones biologiques réalisent facilement un certain nombre d'applications telles que la reconnaissance de formes, le traitement du signal, l'apprentissage par l'exemple, la mémorisation, la généralisation. Ces applications sont pourtant, malgré tous les efforts déployés en algorithmique et en intelligence artificielle, à la limite des possibilités actuelles. C'est à partir de l'hypothèse que le comportement intelligent émerge de la structure et du comportement des éléments de base du cerveau que les réseaux de neurones artificiels se sont développés. Les réseaux de neurones artificiels sont des modèles, à ce titre ils peuvent être décrit par leurs composants, leurs variables descriptives et les interactions des composants.

1 Composant (le neurone artificiel)

1.1 Structure

La figure 1 montre la structure d'un neurone artificiel. Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amonts. A chacune de ces entrées est associée un poids w abréviation de weight (poids en anglais) représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals. A chaque connexion est associée un poids.

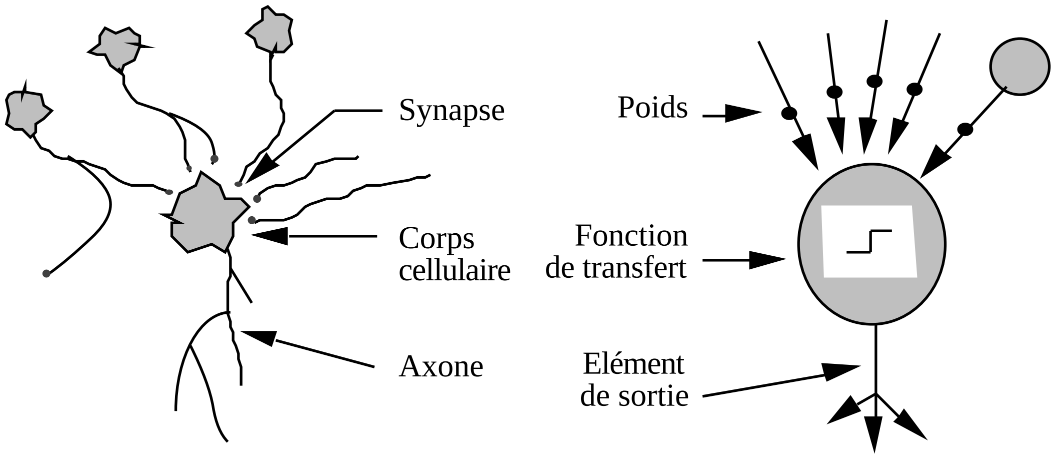


Figure 1. Mise en correspondance neurone biologique / neurone artificiel

La figure 2 donne les notations que nous utilisons dans cet ouvrage.

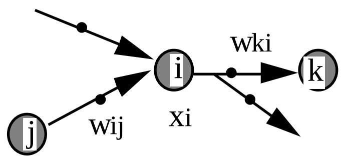


Figure 2. Structure d'un neurone artificiel. Pour le neurone d'indice i, les entrées sur celui-ci sont de poids w_{ij} alors que les connexions avals sont de poids w_{ki}.

1.2 Comportement

On distingue deux phases. La première est habituellement le calcul de la somme pondérée des entrées (a) selon l'expression suivante :

$$a = \sum (w_i \cdot e_i)$$

A partir de cette valeur, une fonction de transfert calcule la valeur de l'état du neurone. C'est cette valeur qui sera transmise aux neurones avals. Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur la figure 3. On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont continues, offrant une infinité de valeurs possibles comprisent dans l'intervalle [0, +1] (ou [-1, +1]).

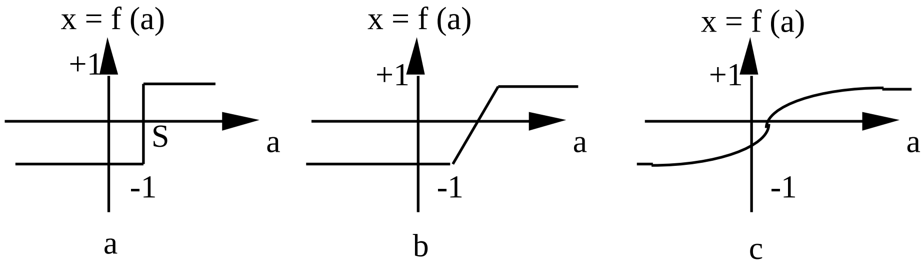


Figure 3. Différents types de fonctions de transfert pour le neurone artificiel,
a : fonction à seuil (S , la valeur du seuil), b : linéaire par morceaux, c : sigmoïde.

Nous constatons que les équations décrivant le comportement des neurones artificiels n'introduisent pas la notion de temps. En effet, et c'est le cas pour la plupart des modèles actuels de réseaux de neurones, nous avons affaire à des modèles à temps discret, synchrone, dont le comportement des composants ne varie pas dans le temps.

2 Variables decriptives

Ces variables décrivent l'état du système. Dans le cas des réseaux de neurones qui sont des systèmes non autonomes, un sous-ensemble des variables descriptives est constitué par les variables d'entrée, variables dont la valeur est déterminée extérieurement au modèle.

3 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

Réseau multicouche (au singulier) : les neurones sont arragés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones

des couches avales (fig. 4). Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelés couches cachées.

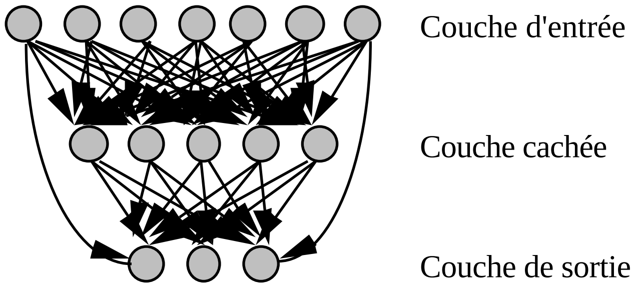


Figure 4. Définition des couches d'un réseau multicouche.

Réseau à connexions locales : Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche avale (fig. 5). Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

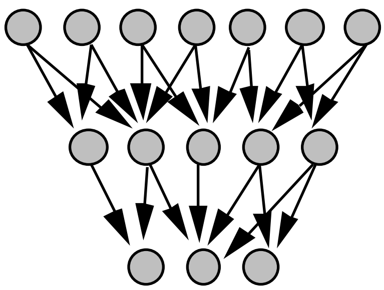


Figure 5. Réseau à connexions locales

Réseau à connexions récurrentes : les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales (fig. 6).

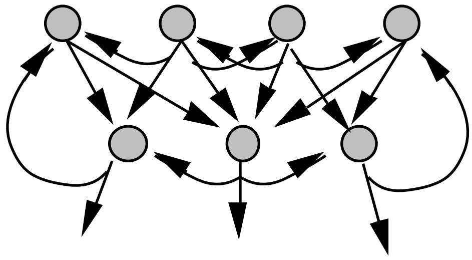


Figure 6. Réseau à connexions récurrentes

Réseau à connexion complète : c'est la structure d'interconnexion la plus générale (fig. 7). Chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

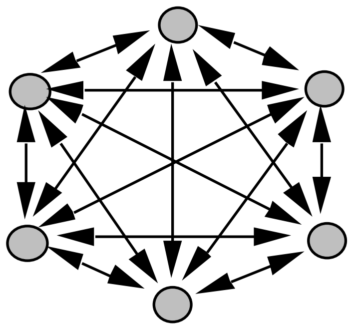


Figure 7. Réseau à connexions complète

Il existe de nombreuse autres topologies possibles, mais elles n'ont pas eu à ce jour la notoriété des quelques unes que nous avons décrites ici.

4 Fonctionnement

4.1 Perceptron

Avant d'aborder le comportement collectif d'un ensemble de neurones, nous allons présenter le Perceptron (un seul neurone) en phase d'utilisation. L'apprentissage ayant été réalisé, les poids sont fixes. Le neurone de la figure 8 réalise une simple somme pondérée de ses entrées, compare une valeur de seuil, et fourni une réponse binaire en sortie. Par exemple, on peut interpréter sa décision comme classe 1 si la valeur de x est +1 et classe 2 si la valeur de x est -1.

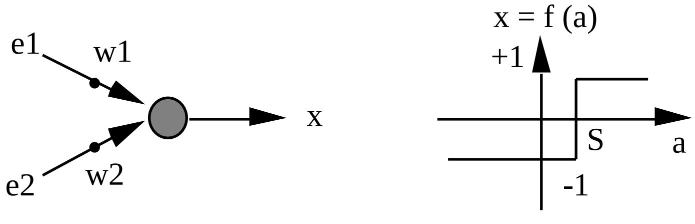


Figure 8. Le Perceptron : structure et comportement. Les connexions des deux entrées e_1 et e_2 au neurone sont pondérées par les poids w_1 et w_2. La valeur de sortie du neurone est notée x. Ele est obtenue après somme pondérée des entrées (a) et comparaison à une valeur de seuil S.

Question : Sachant que les poids du Perceptron à deux entrées sont les suivants : $w_1 = 0.5$, $w_2 = 0.2$ et que la valeur de seuil est S = 0.0, déterminez son comportement, sachant que les comportements du ET logique, OU logique et OU exclusif sont rappelés Table 1 :

ET			OU			OU Exclusif		
e1	e2	x	e1	e2	x	e1	e2	x
1	1	1	1	1	1	1	1	1
1	-1	-1	1	-1	1	1	-1	-1
-1	1	-1	-1	1	1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1

Réponse : OU

4.2 Réseau multicouche en phase d'association

Le comportement collectif d'un ensemble de neurones permet l'émergence de fonctions d'ordre supérieure par rapport à la fonction élémentaire du neurone. Imaginer de prime abord un tel comportement n'est pas facile, nous nous appuyons sur un exemple illustratif et donc réductionniste.

Soit un réseau multicouche composé de 361 (19 x 19), 25 et 361 neurones. Ce réseau a appris à associer à la lettre "a" présentée en entrée la même lettre en sortie (fig.9). Présentons au réseau cette lettre avec quelques erreurs : un certain nombre de pixels ont été inversé (ils sont passés de blanc à noir ou inversement). L'image est composée de 19 x 19 pixels, chacun de ces pixels est associé à un neurone de la couche d'entrée. Chacun des 25 neurones de la couche cachée reçoit 361 connexions (une pour chaque neurone d'entrée) et envoie sa sortie à chacun des neurones de la couche de sortie (au nombre de 361). Dans notre exemple, la couche cachée se compose de 25 neurones, mais ce nombre, à la différence des couches d'entrée et de sortie, n'est pas impératif. Il y a donc 2 . (361 . 25) = 18050 connexions dans le réseau.

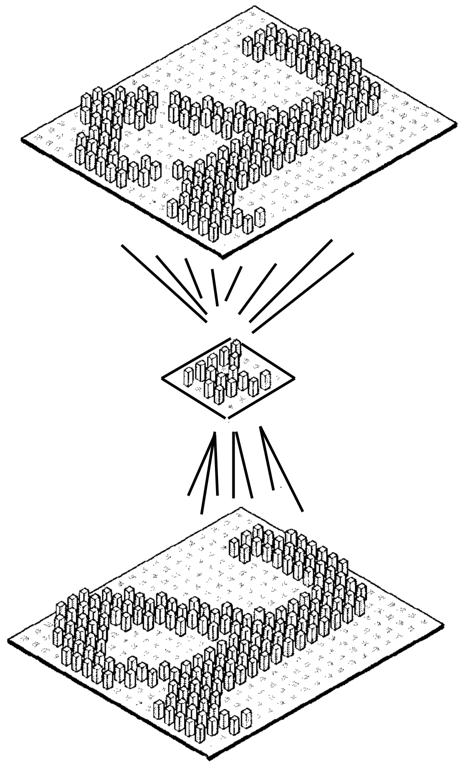


Figure 9. Comportement en phase de reconnaissance d'un réseau de neurone multicouche lors d'un tâche d'auto-association. Les neurones sont binaires. La valeur d'activation de chaque neurone est indiquée par la hauteur de la colonne. Les neurones sont rangés par couche, tous les neurones d'une couche sont connextés à tous les neurones de la couche suivante (avale).

La première étape code l'image d'entrée sur le réseau. Il s'agit pour chaque neurone de la couche d'entrée de fixer la valeur de son état selon la couleur du pixel correspondant. Si les neurones qui composent le réseau sont binaires, on choisit arbitrairement de coder un pixel noir par un niveau d'activation du neurone égal à 1 ; si le pixel est blanc alors le niveau d'activation du neurone correspondant est égal à 0.

La seconde étape est celle du calcul de la réponse du réseau qui se décompose en autant de sous-étapes qu'il y a de couches dans le réseau. Ainsi, chaque neurone de la couche d'entrée envoie sa valeur aux neurones de la couche cachée. Chacun des neurones de la couche cachées est en fait un Perceptron à 361 entrées. Chacun des neurones réalise la somme pondérée de ses entrées et seuille. Ce processus est effectué en parallèle et indépendamment pour tous les neurones de la couche cachée. Lorsque le vecteur d'activation de la couche cachée a été obtenu, le même processus est répété avec les neurones de la couche de sortie. On considère ceux-ci comme 361 Perceptrons indépendants à 25 entrées.

La dernière étape est l'interprétation du vecteur d'activation de la couche de sortie par l'expérimentateur. Dans notre cas, on réalise l'opération inverse du codage initial, à savoir

associer un pixel noir à chacun des neurones de la couche de sortie dont la valeur d'activation est égal à 1, un pixel blanc dans le cas contraire.

Dans la figure 9, il faut remarquer que si les vecteurs d'activation des couches d'entrée et de sortie semblent directement interprétables, il n'en est rien en ce qui concerne la couche cachée. Lorsque les neurones qui composent le réseau sont à valeur continue, les possiblités offertes sont plus nombreuses. L'image d'entrée peut être composé de plusieurs niveaux de gris. On associe alors arbitrairement à chaque niveau de gris un niveau d'activation du neurone spécifique. Le calcul du vecteur d'activation de la couche cachée reste identique dans son principe avec cependant le fait que l'état de chaque neurone n'est plus binaire. L'interprétation de la réponse du réseau fourni une image composée de niveaux de gris.

4.3 Réseau à connexion complète

Chaque vecteur d'activation représente la réponse du réseau à une date particulière. Pour faciliter la représentation, nous avons déplié dans l'espace les évolutions temporelles du réseau à connexion complète (trois cycles fig. 10). D'un instant au suivant, chaque neurone recalcule indépedemment et en parallèle son état. Rappelons que chacun des neurones est connecté à tous les autres, ce qui implique pour chacun d'entre eux de recevoir 361 connexions et d'envoyer sa sortie sur ses 361 voisins. La principale différence entre les évolutions temporelles d'un réseau à connexion complète et le calcul de la sortie dans un réseau multicouche est que pour le premier les poids des connexions entre deux évolutions temporelles sont identiques, alors que pour le second, d'une couche à l'autre les poids des connexions sont différents.

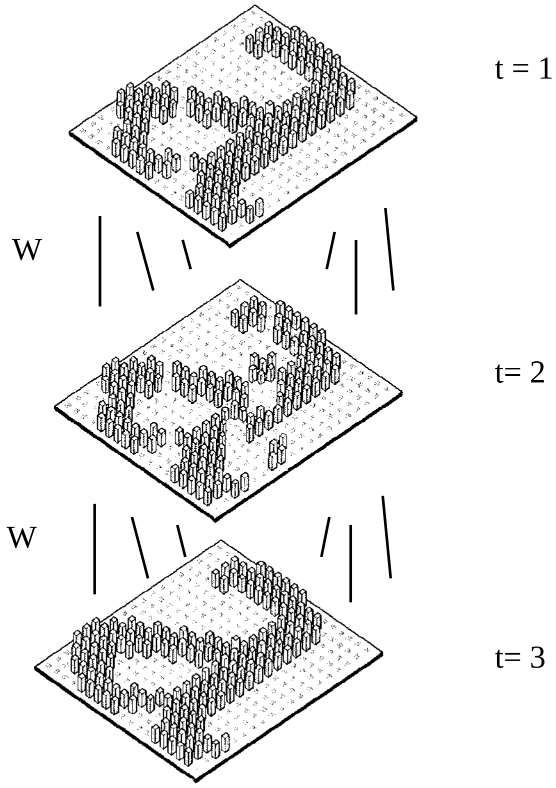
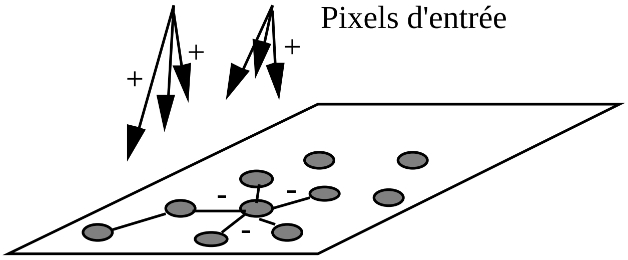


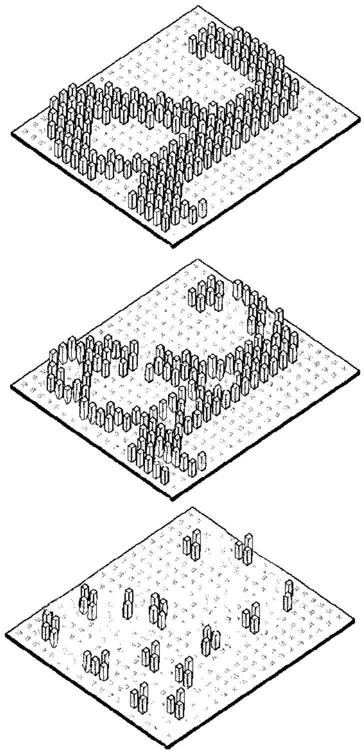
Figure 10. Evolution du vecteur d'activation d'un réseau à connexion complète sur une période de trois cycles. La matrice des poids W est complète (361 x 361 = 130321 poids). Entre deux "couches", c'est la même matrice de poids.

4.4 Réseau à inhibition latérale récurrente.

Les poids sont fixés a priori lors de la construction du réseau, il n'y a pas de phase d'apprentissage. La structure du réseau est représentée sur la figure 11. Il y a une seule couche de neurones. Les connexions sont localisées, chaque pixel d'entrée est en relation (excitatrice) avec un nombre réduit de neurones. De plus, on remarque la présence de connexions récurrentes inhibitrices localisées autour de chaque neurone.



b



t= 1

t=3

t=6

Figure 11. Réseau à inhibition latérale récurrente.

a) Architecture, chaque pixel d'entrée est connecté à un ensemble de neurones par des connexions excitatrices. Sur le réseau, chaque neurone réalise des connexions locales inhibitrices.

b) Comportement, seuls les pixels noirs de la forme d'entrée envoient une information sur le réseau. Pour chaque neurone, il y a compétition entre l'excitation en provenance de l'entrée et l'inhibition en provenance de ses voisins. On observe alors le développement au cours du temps de sous-groupes d'activité significatifs (angles, intersections, extrémités, etc).

En utilisation, une image est présentée en entrée. Elle n'est pas codée comme précedemment : un groupe de neurones est associé chaque pixel dont la couleur détermine le niveau d'activation. De plus, à la différence d'un réseau multicouche classique, la réponse n'est obtenue qu'aprés stabilisation de l'état de sortie. Le régime transitoire est du au retour d'information depuis les neurones voisins. De fait, cette boucle doit être réalisée un certain nombre de fois avant que l'on obtienne une valeur fixe en sortie.

5 Conclusion

Gràce aux quelques exemples de comportements vus, il est facile de comprendre que la disparition d'un ou même de plusieurs neurones (ou de connexions) ne provoque pas une rupture brutale du traitement. En fait, la dégradation du comportement est fonction de la quantité d'éléments détruits. Cette propriété est désignée sous le terme de résistance aux pannes.

Par rapport aux données biologiques recencées au chapitre précédent, nous constatons :

- une réduction du nombre de connexions par neurone (de 10.000 à quelques centaines maximum),
- une réduction drastique du nombre de neurones pour un réseau artificiel (quelques centaines à comparer aux mille milliards du cerveau),
- une diminution de la complexité de la synapse et l'atypie des topologies proposées.

La plupart des modèles que nous allons découvrir sont des modèles synchrones à temps discrets et combinatoires, alors que le monde biologique est asynchrone et continu. Il est important de noter que la nature du message nerveux biologique (codage en fréquence) devient désormais combinatoire (codage spatial). Nous constatons que la complexité biologique n'est pas conservée.

6 Comportements combinatoire et séquentiel (TD)

Parmi les variables descriptives qui ne sont pas des variables d'entrée, on appelle variables d'état les variables dont la valeur à n'importe quelle date, ajoutée à la valeur des entrées, déterminent de manière unique les valeurs de toutes les autres. Les états caractérisent les possibilités de mémorisation du système : l'information qu'il peut retenir des stimuli passés et qui modifiera la réponse à des stimuli futurs. Un système est sans mémoire s'il n'y a pas de variables d'état. Pour un tel système, les réponses présentes et futures ne peuvent en aucune manière être affectées par des stimuli passés. De tels systèmes sont appelés combinatoires, car leur réponse à n'importe quelle date est uniquement fonction du stimulus reçu à cet instant.
Question : Donnez l'équation décrivant le comportement de ces systèmes.
Reponse : Le comportement de ces systèmes est défini par l'équation (F est la fonction réalisée, E(t) est l'entrée, S(t) est la sortie) :

$$S(t) = F(E(t))$$

Un grand nombre de modèles neuronaux, parmi les plus utilisés, n'ont pas de variables d'état et montrent donc un comportement combinatoire : réseau multicouche, carte auto-organisatrice, réseau ART1, ...

Inversement, un système à états répondra différemment à des entrées présentes et identiques selon l'histoire de ses stimuli d'entrées passés.
Question : Son comportement, nommé comportement séquentiel, est donc défini par une équation de la forme ?

Reponse : $S(t) = F(E(t), E(t-1), E(t-2), ..., E(1), E(0))$

Un petit nombre de modèles connexionnistes montrent un comportement séquentiel, nous y revenons au chapitre Questions récapitulatives.

4 Apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

Définition :

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage. Le modèle sans apprentissage présente en effet peu d'intérêt. Dans la majorité des algorithmes actuels, les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Certains modèles de réseaux sont improprement dénommés à apprentissage permanent. Dans ce cas il est vrai que l'apprentissage ne s'arrête jamais, cependant on peut toujours distinguer une phase d'apprentissage (en fait de remise à jour du comportement) et une phase d'utilisation. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées.

Au niveau des algorithmes d'apprentissage, il a été défini deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé. Cette distinction repose sur la forme des exemples d'apprentissage. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (Entrée, Sortie associée) alors que l'on ne dispose que des valeurs (Entrée) pour l'apprentissage non supervisé. Remarquons cependant que les modèles à apprentissage non supervisé nécessite avant la phase d'utilisation une étape de labélisation effectuée l'opérateur, qui n'est pas autre chose qu'une part de supervision.

1 La loi de Hebb, un exemple d'apprentissage non supervisé

La loi de Hebb (1949) s'applique aux connexions entre neurones, comme le représente la figure 1.

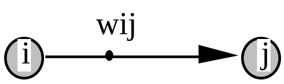


Figure 1. i le neurone amont, j le neurone aval et w_{ij} le poids de la connexion.

Elle s'exprime de la façon suivante

"*Si 2 cellules sont activées en même temps alors la force de la connexion augmente*".

La modification de poids dépend de la coactivation des neurones présynaptique et post synaptique, ainsi que le montre la table 1. x_i et x_j sont respectivement les valeurs d'activation des neurones i et j, ∂wij (dérivée partielle du poids) correspond à la modification de poids réalisée.

x_i	x_j	∂w_{ij}
0	0	0
0	1	0
1	0	0
1	1	+

Table 1. La loi de Hebb.

La loi de Hebb peut être modélisée par les équations suivantes (w(t+1) est le nouveau poids, wij(t) l'ancien) :

$w_{ij}(t+1) = w_{ij}(t) +\partial w_{ij}(t)$

$\partial w_{ij}(t) = x_i \, . \, x_j$ (la coactivité est modélisée comme le produit des deux valeurs d'activation)

L'algorithme d'apprentisage modifie de façon itérative (petit à petit) les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement.

1/ Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.

2/ Présentation d'une entrée $E_l = (e_1, \dots e_n)$ de la base d'apprentissage.

3/ Calcul de la sortie obtenue x pour cette entrée :

$a = \sum (w_i \, . \, e_i) - S$ (la valeur de seuil est introduite ici dans le calcul de la somme pondérée)

x = signe (a) (si $a > 0$ alors $x = +1$ sinon $a \leq 0$ alors $x = -1$)

4/ Si la sortie x est différente de la sortie désirée d_l pour cet exemple d'entrée E_l alors modification des poids (µ est une constante positive, qui spécifie le pas de modification des poids) :

$w_{ij}(t+1) = w_{ij}(t) + \mu.(x_i \, . \, x_j)$

5/ Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

Exemple d'application de l'algorithme d'apprentissage de Hebb :

Choisissons pour les neurones un comportement binaire. Les entrées e_1 et e_2 sont considérées comme des neurones (fig. 2).

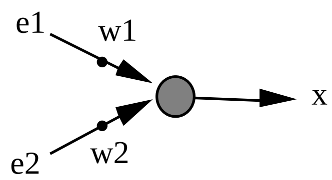


Figure 2. Réseau de 3 neurones (les 2 entrées sont considérées comme deux neurones) pour la résolution du problème exprimé table 2.

Nous allons réaliser l'apprentissage sur un problème très simple. La base d'apprentissage est décrite par la table 2 :

e_1	e_2	x	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

Table 2. Base d'exemples d'apprentissage pour la loi de Hebb.

1/ Conditions initiales : $\mu = +1$, les poids et le seuil sont nuls.
2/ Calculons la valeur de x pour l'exemple (1) :
3/ $a = w_1.e_1 + w_2.e_2 - S = 0.0 . 1 + 0.0 . 1 - 0.0 = 0$ $a \leq 0 \Rightarrow x = -1$
4/ La sortie est fausse, il faut donc modifier les poids en applicant :
$w_1 = w_1 + e_1.x = 0.0 + 1.1 = 1$
$w_2 = w_2 + e_2.x = 0.0 + 1.1 = 1$
2/ On passe à l'exemple suivant (2) :
3/ $a = 1.1 + 1.-1 -0.0 = 0$ $a \leq 0 \Rightarrow x = -1$
4/ La sortie est fausse, il faut donc modifier les poids en applicant :
$w_1 = 1 + 1.1 = 2$
$w_2 = 1 + 1.-1 = 0$
…/ L'exemple suivant (3) est correctement traité : a = -2 et x = -1 (la sortie est bonne).
On passe directement, sans modification des poids à l'exemple (4). Celui-ci aussi est correctement traité. On revient alors au début de la base d'apprentissage : l'exemple (1). Il est correctement traité, ainsi que le second (2). L'algorithme d'apprentissage est alors terminé : toute la base d'apprentissage a été passée en revue sans modification des poids.

Question : Soit le réseau composé de 4 neurones d'entrée et d'un neurone de sortie ($w_1 = w_2 = w_3 = w_4 = S = 0$) et la base d'apprentissage :

	e_1	e_2	e_3	e_4	x
	1	-1	1	-1	1
1	1	1	1	1	
1	1	1	-1	-1	
1	-1	-1	1	-1	

Recherchez les valeurs de poids qui résolvent le problème.

Réponse : Cet algorithme d'apprentissage ne permet pas de trouver une solution à ce problème. Nous ne sommes capables d'exprimer une combinaison des activations en corrélation avec la sortie. Pourtant, il existe des solutions comme par exemple (w_1 = -0.2, w_2 = -0.2, w_3 = 0.6, w_4 = 0.2). Un algorithme de calcul efficace pour ce problème est l'apprentissage sur le modèle du Perceptron abordé au chapitre suivant.

Remarque : Il existe une possibilité de calculer les valeurs des connexions à partir des exemples, sans utiliser l'algorithme itératif. Si l'on initialise les poids à zéro et que l'on présente les exemples de la base d'apprentissage, la valeurs des poids à l'issue de l'apprentissage est :

$w_{ij} = \sum_l x_{il} \cdot x_{jl}$ où l est l'indice de l'exemple dans la base d'apprentissage

2 La règle d'apprentissage du Perceptron, un exemple d'apprentissage supervisé

La règle de Hebb ne s'applique pas dans certain cas, bien qu'une solution existe (cf exercice du paragraphe précédent). Un autre algorithme d'apprentissage a donc été proposé, qui tient compte de l'erreur observée en sortie.

L'algorithme d'apprentissage du Perceptron est semblable à celui utilisé pour la loi de Hebb. Les différences se situent au niveau de la modification des poids.

1/ Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.

2/ Présentation d'une entrée $E_l = (e_1, \dots e_n)$ de la base d'apprentissage.

3/ Calcul de la sortie obtenue x pour cette entrée :

$a = \sum (w_i \cdot e_i) - S$

x = signe (a) (si a > 0 alors x = +1 sinon a ≤ 0 alors x = -1)

4/ Si la sortie x du Perceptron est différente de la sortie désirée d_l pour cet exemple d'entrée E_l alors modification des poids (µ le pas de modification) :

$w_i(t+1) = w_i(t) + \mu.((d_l - x).e_i)$

Rappel : d_l = +1 si E est de la classe 1, d_l = -1 si E est de la classe 2 et (d_l - x) est une estimation de l'erreur.

5/ Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron :

Base d'exemples d'apprentissage :

e_1	e_2	d	
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

1/ Conditions initiales : w_1 = -0.2, w_2 = +0.1, S = 0, (μ = +0.1)

2/ a(1) = -0.2 + 0.1 -0.2 = -0.3

3/ x(1) = -1 (la sortie désirée d(1) = +1, d'où modification des poids)

4/ w_1 = -0.2 + 0.1 . (1 + 1) . (+1) = 0

w_2 = +0.1 + 0.1 . (1 + 1) . (+1) = +0.3

2/ a(2) = +0.3 - 0.2 = +0.1

3/ x(2) = +1 Faux

4/ w_1 = 0 + 0.1 . (-1 - 1) . (-1) = +0.2

w_2 = +0.3 + 0.1 . (-1 - 1) . (+1) = +0.1

2-3/ a(3) = -0.2 -0.1 -0.2 = -0.5 Ok

2-3/ a(4) = +0.2 - 0.1 - 0.2 = -0.1 Ok

2-3/ a(1) = +0.2 + 0.1 - 0.2 = +0.1 Ok

2-3/ a(2) = -0.2 + 0.1 - 0.2 = -0.1 Ok

5/ Tous les exemples de la base ont été correctement traités, l'apprentissage est terminé.

Le Perceptron réalise une partition de son espace d'entrée en 2 classes (1 et 2) selon la valeur de sa sortie (+1 ou -1). La séparation de ces deux zones est effectuée par un hyperplan (fig. 3). L'équation de la droite séparatrice est :

$$w_1.e_1 + w_2.e_2 - S = 0$$

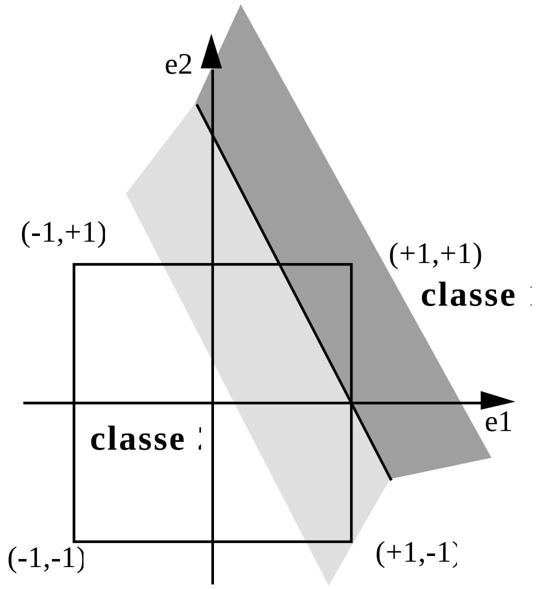


Figure 3. Partition de l'espace d'entrée de dimension 2 réalisée par un Perceptron se comportant comme un ET booléen. Les 4 exemples de la base d'apprentissage sont les 4 arêtes du carré. Les paramètres du Perceptron sont : $w_1 = 0.2$, $w_2 = 0.1$ et $S = -0.2$.

Remarque : si les exemples d'apprentissage étaient différents, par exemple représentant le OU, alors c'est le comportement du OU qui aurait été appris avec le *même* algorithme d'apprentissage. D'autre part, il est possible de considéré que le seuil S est le poids d'une connexion dont le neurone amont est toujours dans l'état -1. La valeur de seuil se modifie alors en même temps que les autres poids. La convergence de l'algorithme vers la solution est plus rapide (si cette solution existe). On appelle itération d'apprentissage, le passage de tous les exemples de la base d'apprentissage une fois dans l'algorithme.

Question : Reprendre la question précédente (apprentissage non supervisé) et la résoudre en applicant l'apprentissage du Perceptron. On ne modifiera pas le seuil S dans cet exemple précis.
Réponse : $w_1 = -0.2$, $w_2 = -0.2$, $w_3 = 0.6$, $w_4 = 0.2$

3 TP Perceptron

1/ Simuler la fonction ET avec des poids fixes et sans apprentissage. Les paramètres du Perceptron sont les suivants : $w_1 = 0.2$, $w_2 = 0.1$ et $S = -0.2$.

Les exemples de comportement à vérifier (ET) sont rappelés sur la table suivante :

e_1	e_2	d	
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)

1	-1	-1	(4)

2/ Pour la même base d'apprentissage, réaliser l'apprentissage (ne pas oublier la modification du seuil). Le choix des conditions initiales est confié au hasard. Dans un première étape, il est conseillé de refaire pas à pas l'exemple de cet ouvrage : w_1 = -0.2, w_2 = +0.1, S = 0, μ = +0.1 (Conditions initiales). Puis faites varier μ.

3/ Essayer d'apprendre le XOR.

e_1	e_2	d	
1	1	1	(1)
-1	1	-1	(2)
-1	-1	1	(3)
1	-1	-1	(4)

4/ Sachant que le XOR peut s'écrire comme : ((e_1 ET (NON(e_2))) OU (NON(e_1) ET e_2)) proposez une solution pour réaliser le XOR avec 3 Perceptrons. (NON(1) = -1 et inversement)

e_1	e_2	d	
1	1	1	(1)
-1	1	1	(2)
-1	-1	-1	(3)
1	-1	1	(4)

Table du OU

L'apprentissage de chacun des Perceptrons est réalisé séparement des autres. Qu'en déduisez-vous quant aux possibilités d'un Perceptron ? d'une association de Perceptrons ?

5/ Réaliser la fonction ET et OU avec 2 neurones. Dans ce cas, le réseau se compose de 2 entrées, 2 neurones et 4 poids. L'apprentissage de chacune des fonctions n'est pas séparé. Il faut donc construire une base d'apprentissage spécifique de ce problème (qui ne comprendra pas plus de 4 exemples).

6/ Reconnaissance de caractère avec un Perceptron. Un caractère est codé sur 4 x 7 = 28 pixels. Il y donc 28 entrées sur le perceptron. Tester la généralisation et la résistance au bruit en proposant à l'issu de l'apprentissage des caractères "abimés".

7/ Reconnaissance de caractères : on associe a priori un caractère par neurone. Il faut donc autant de Perceptrons que de caractères à reconnaitre. Tester la généralisation. Etudier les erreurs, sur quels caractères apparaissent-elles, comment peut-on y remédier ?

On prendra soin de cette construction de la base de caractères, qui est aussi utilisée dans les TP relatifs aux mémoires associatives, cartes auto-organisatrices, réseaux ART, réseaux multicouches.

5 Mémoires associatives

Les mémoires associatives ont été proposés par plusieurs auteurs dès 1977 dont T. Kohonen. Nous nous appuyons sur ses travaux qui ont été admirablement résumé par C. Jutten. Dans mémoire associative, le terme "mémoire" fait référence à la fonction de stockage de l'information et le terme "associative" au mode d'adressage. L'expression "mémoire adressable par son contenu" est aussi souvent employée. L'information mémorisée ne peut être obtenue à une adresse précise, le seul moyen d'accés est de fournir une information. Dans le cas des mémoires auto-associatives, il faut fournir tout ou partie de l'information mémorisée. Ces mémoires sont donc principalement utilisées pour la reconstruction de données : l'opérateur fourni une information partielle que le système complète. Des expérimentation dans ce sens ont été faite avec l'annuaire électronique où l'utilisateur tape le maximum d'informations relatives à sa demande, que le système complète et corrige (fig. 1). Les mémoires hétéro-associatives se différencient des précédentes en rendant une information différente. Par exemple, si la clef d'entrée est une image de visage, le système répond par le nom de la personne correspondante.

Appris :	Jean Dupond, 22 rue du 29 Février, 99001 Asnières, 66 38 70 29
Clef :	Jean Dupont, rue du 29 Septembre, Asnières,
Résultat :	Jean Dupond, 22 rue du 29 Février, 92501 Asnières, 66 38 70 29

Figure 1. Exemples d'interprétations (et de corrections) de requêtes d'un utilisateur de l'annuaire électronique par une mémoire auto-associative (il peut subsister des erreurs).

1 Structure

La structure neuronale d'une mémoire associative est similaire à celle d'une carte auto-organisatrice sans la notion de voisinage (cf chapitre suivant), ou à celle d'un ensemble de Perceptrons tous alimentés par les mêmes entrées. La figure 1 montre cette architecture où chaque entrée est connectée par des poids modifiables à toutes les sorties. La dimension de la couche d'entrée est de n neurones, celle de sortie de p. Il y a donc n.p poids dans ce réseau.

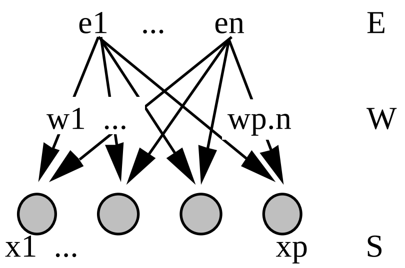


Figure 1. Structure d'une mémoire associative

2 Fonctionnement

Le principe de fonctionnement d'une mémoire associative se résume ainsi. Soit (E_1, E_2, …, E_l, …) un ensemble de vecteurs de R^n. A chaque vecteur E_l appelé "prototype" de l'espace d'entrée est associé un vecteur de sortie S_l. La relation d'association entre E_l et S_l est linéaire. Elle est donnée par l'équation :

$$S_l = W . E_l$$

où W est la matrice des poids de dimension (p.n). C'est une matrice rectangulaire de p lignes et n colonnes.

L'objectif est de faire réaliser à ce réseau des associations entre les vecteurs d'entrées et les vecteurs de sortie désirés. Ceci nécessite une étape d'apprentissage.

3 Apprentissage

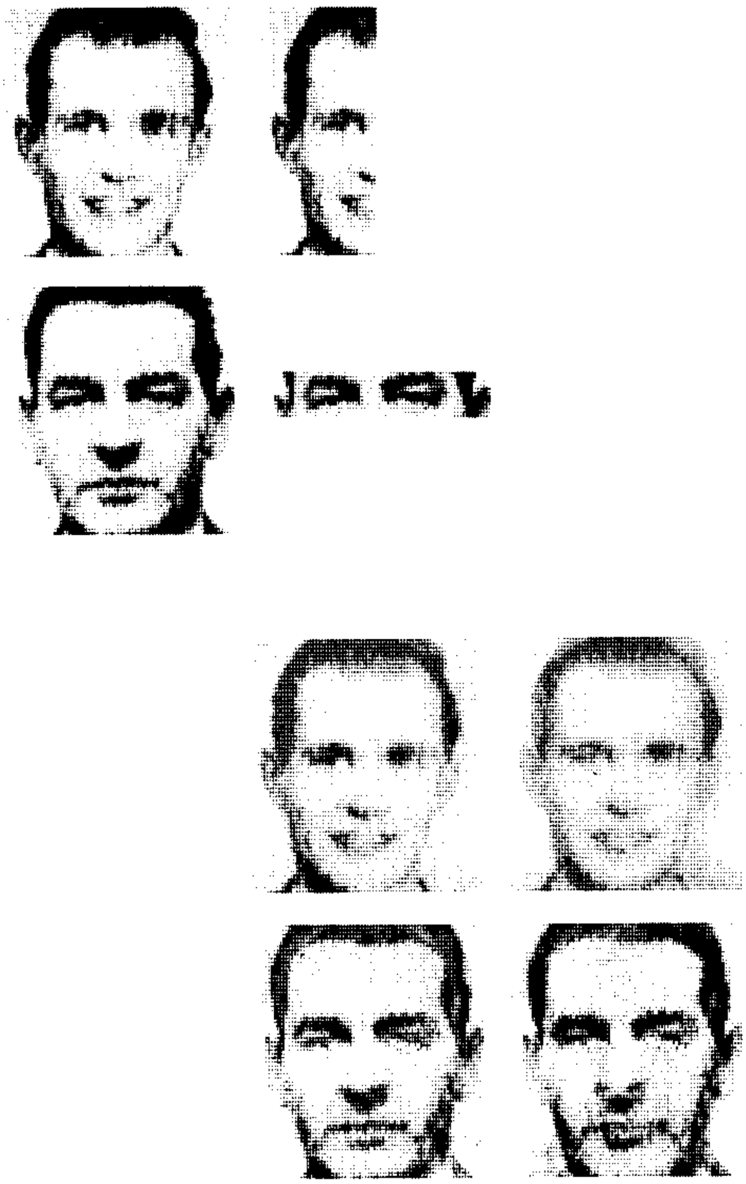
L'apprentissage est de type supervisé. La base d'apprentissage est composée de couple de vecteurs d'entrée et des vecteurs de sortie associés. L'algorithme d'apprentissage initial fait appel à la règle de Hebb. Une entrée E_l est appliquée sur les neurones d'entrée du réseau et l'on force dans le même temps les valeurs des neurones de sortie à S_l. Les poids de chaque connexion est alors modifié selon la coactivité du neurone afférent (entrée) et du neurone efférent (sortie). Cet algorithme est itéré sur tous les exemples de la base d'apprentissage. A la fin du processus d'apprentissage, si la matrice W est initialement nulle (W = 0), on obtient :

$$W = \sum_l S_l . E_l^T$$

où E_l^T est la transposée du vecteur E_l (qui transforme un vecteur ligne en un vecteur colonne et réciproquement)

Cette expression est en fait un raccourci mathématique au processus d'apprentissage itératif mettant en jeu une règle locale de modification des poids.

4 Résultats



a b c d

Figure 2. Illustration du fonctionnement d'une mémoire auto-associative (d'après Kohonen).

a) Images originales apprises à gauche.

b) Clefs soumises en entrée au réseau.

c) Images restituées par le réseau lorsque 160 images ont été stockées.

d) Images restituées par le réseau lorsque 500 images ont été stockées.

5 TP Mémoires associatives

1/ Ecrire le programme d'une mémoire associative, sachant que les prototypes sont les caractères construits lors du TP Perceptron (4 x 7 pixels).

2/ Tester la généralisation et la résistance au bruit en proposant, à l'issu de l'apprentissage, des caractères "abimés".

3/ Tester les capacités de mémorisation de ce réseau en augmentant la taille de la base d'apprentissage (environ 15% du nombre de neurones dans le réseau).

4/ Donner quelques explications relatives à la génération des erreurs par le système, sur quels caractères apparaissent-elles, comment peut-on y remédier ? (notion d'orthogonalité au sein de la base d'exemples)

6 Carte auto-organisatrice

Ce modèle de carte auto-organisatrice appartient à la classe des réseaux à compétition. Les neurones de la couche de sortie entrent en compétition, de telle façon qu'habituellement, un seul neurone de sortie est activé pour une entrée donnée. Cette compétition entre les neurones est réalisée grâce à des connexions latérales inhibitrices. Nous présentons deux modèles parmi les plus intéressants : la carte auto-organisatrice et le réseau ART1 (au chapitre suivant). Il faut noter que tous deux sont issus de reflexions neuromimétiques : ils se placent originellement comme des modélisation de processus biologiques. Ils ont depuis été récupérés par les ingénieurs connexionnistes comme le montre les applications présentées.

Les cartes auto-organisatrices sont depuis longtemps (!) connues (1977), mais ce n'est que très récemment (1990) que des applications les utilisent : carte phonétique, diagnostic de pannes, compression d'images, robotique, etc. Ces cartes s'organisent par rapport aux exemples d'entrée présentés en respectant les contraintes topologiques de l'espace d'entrée. Il y a mise en correspondance de l'espace d'entrée avec l'espace du réseau. Les zones voisines de l'espace d'entrée sont voisines sur la carte auto-organisatrice (fig. 1)

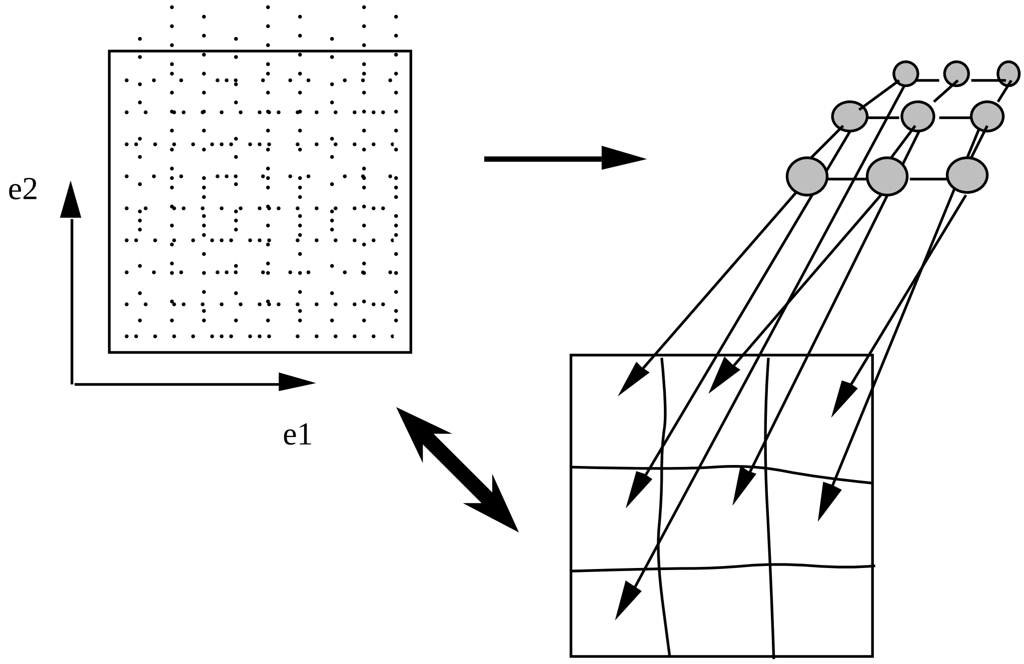


Figure 1. Mise en correspondance de l'espace d'entrée avec l'espace du réseau. L'espace d'entrée avec les exemples (points) d'apprentissage est représenté à gauche. A l'issue de l'apprentissage, chacun des 9 neurones de la carte auto-organisatrice correspond à une zone de l'espace d'entrée (aussi nommée champ récepteur) en bas à droite. Tout point tiré dans un champ récepteur active le neurone correspondant et lui seul.

1 Structure

La figure 2 décrit le réseau organisé en une couche à deux dimensions. Chaque neurone Nk est connecté à un nombre n d'entrées au travers de n connexions plastiques de poids respectifs w. Il existe aussi des connexions latérales de poids fixes, excitatrices dans un proche voisinage.

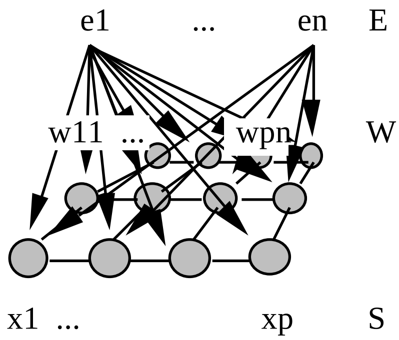


Figure 2. Architecture d'une carte auto-organisatrice (réseau 2D).
Chaque neurone est connecté à ses 4 plus proches voisins. Ces connexions sont de poids fixes. Tous les neurones sont connectés aux entrées par des connexions plastiques.

2 Fonctionnement

A la présentation d'un eentrée, un neurone sur la carte est sélectionné. Il correspond le plus possible à cette entrée (minimisation d'une distance). On peut ainsi réaliser des classifications ou de la reconnaissance de formes. Le modèle de réseau neuronal proposé par Kohonen montre des propriétés d'auto-organisation et de représentation topologique de l'espace d'entrée (espace afférent).

3 Apprentissage

La loi de modification des poids des connexions (poids synaptiques) est dérivée de celle de Hebb. Dans le cas où les exemples d'entrées sont des vecteurs à deux composantes, l'algorithme d'apprentissage est le suivant :

1/ Initialisation des poids à des valeurs aléatoires.
2/ Présentation d'une entrée $E_l = (e_1, e_2)$.
3/ Calcul de la distance de chacun des neurones par rapport à e_1 et e_2

$x_j = |w_{j1} - e_1| + |w_{j2} - e_2|$

4/ Sélection du neurone le plus proche : $Min(x) = x_i$
5) Modification des poids pour le neurone choisi (i) et ses 4 plus proches voisins (k).
µ et ß sont deux paramètres correspondant au pas de modification des poids. µ pour le neurone choisi et ß pour ceux du voisinnage.

$w_{i1} = w_{i1} + \mu . (e_1 - w_{i1})$

$w_{i2} = w_{i2} + \mu . (e_2 - w_{i2})$

$w_{k1} = w_{k1} + ß. (e_1 - w_{k1})$
$w_{k2} = w_{k2} + ß. (e_2 - w_{k2})$

6) Tant que les performances sont insuffisantes : Retour à l'étape 2 et sélection de l'exemple suivant dans la base d'apprentissage.

Remarque : Cet algorithme est un raccourci mathématique. Originellement, le modèle est biologiquement plus plausible et aussi plus complexe. L'unité de traitement n'est pas le neurone mais la colonne corticale (ensemble d'environ 200 neurones). Un voisinage est défini autour de chaque colonne corticale. Son action est à la fois excitatrice dans un proche voisinage et inhibitrice dans un voisinage plus lointain (fig. 3 et 4) :

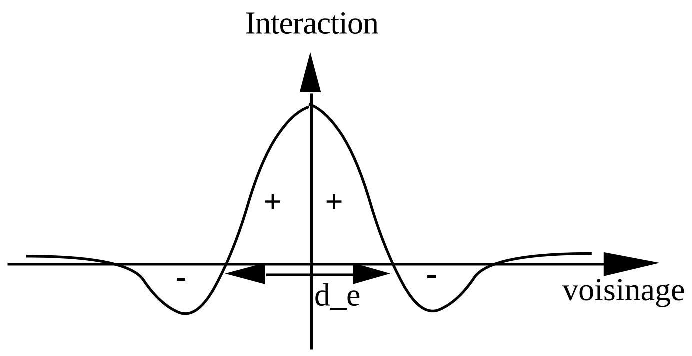


Figure 3. Influence d'un neurone sur ses voisins en fonction de l'éloignement.
+ : excitatrice (w > 0), - : inhibitrice (w > 0), d_e : taille du voisinage excitateur.

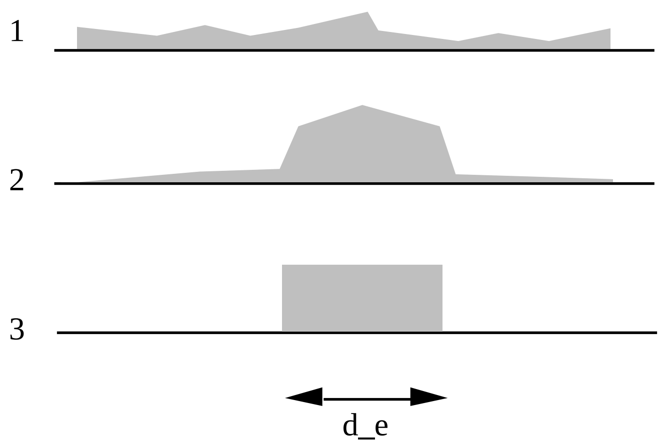
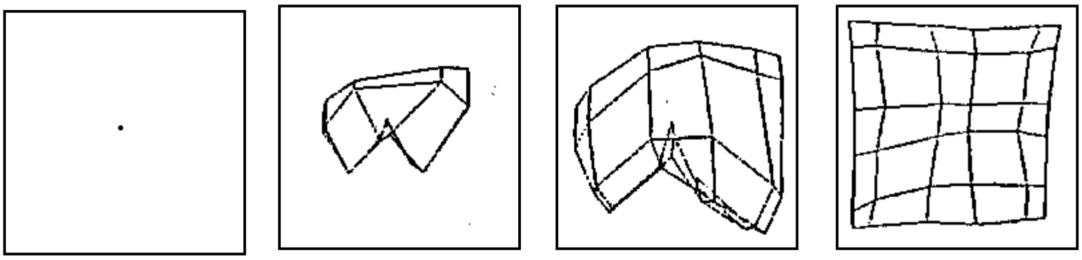


Figure 4. Fonctionnement au cours du temps du réseau. On voit se dégager progressivement, pour une même information, un foyer d'excitation alors que les autres neurones sont inhibés.

4 Résultats

Les informations reçues par le réseau détermine un arrangement spatial optimal des neurones. Les figures graphiques obtenues peuvent être lues comme représentant pour chaque neurone l'endroit du monde extérieur sur l'espace des poids pour lequel son activité est maximale (sélectivité de position).



1 40 200 1000

Figure 5. Le réseau apprend l'espace carré à partir d'un état initial caractérisé par des poids répartis aléatoirement autour d'une valeur centrale. Chaque point est défini par un couple (e_1, e_2). Les liens de voisinage topologiques sont matérialisés par des traits reliant chaque point. Sous le graphe figure le nombre d'itérations correspondant au nombre de points utilisés pour l'apprentissage (d'aprés Y. Coiton).

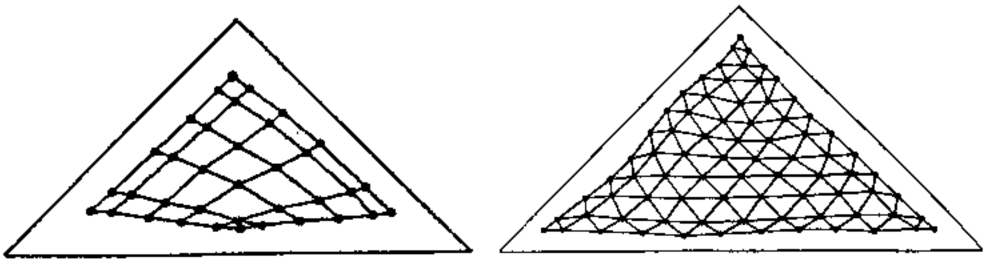
La figure 6 illustre la propriété d'arrangement spatial optimal. Le réseau est ici à une dimension (seulement deux voisins) et l'espace des entrées est à deux dimensions.



a b

Figure 6. Arrangement spatial optimal pour un réseau une dimension,
a) dans un carré, b) dans un rectangle.

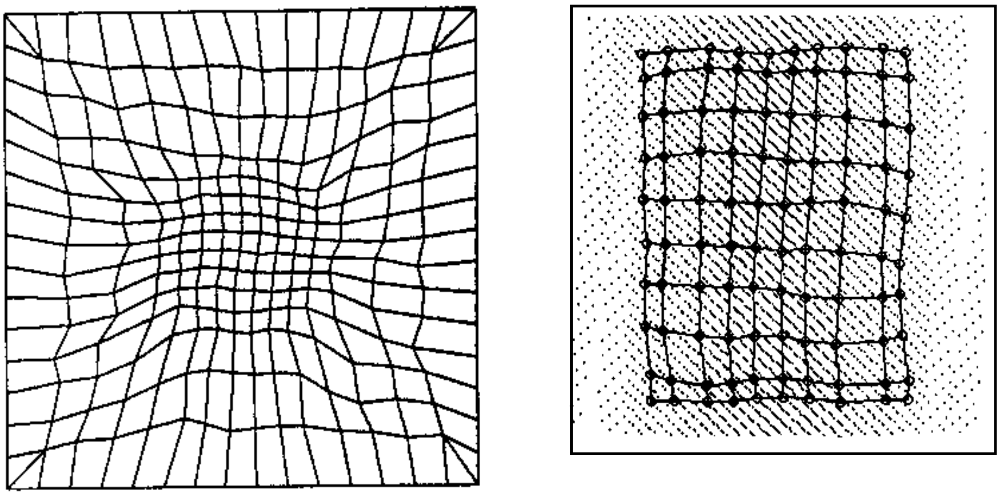
Illustration de l'adéquation entre la dimension et la forme du réseau avec l'espace des entrées. L'exemple choisi est celui d'une carte triangulaire équilatérale (chaque côté du triangle comporte le même nombre de neurones). Hormis sur la périphérie, chaque neurone possède six voisins.



a b

Figure 7. Illustration de l'adéquation entre la forme de la carte et la forme de l'espace.
a) Carte carré dans un triangle, b) Carte triangulaire dans le même triangle.

L'arrangement spatial optimal; est une autre propriété remarquable des cartes, qui s'organisent de façon à approximer la fonction densité de probabilité des vecteurs d'entrée. Nous présentons deux exemples d'occupation non uniforme d'un espace carré par un réseau carré (figure 8), selon la distribution des points tirés durant l'apprentissage.



a b

Figure 8. Occupation non uniforme d'un carré par un réseau carré.
Les neurones se concentrent dans la zone de distribution plus élevée.
a) Le centre de l'espace est beaucoup plus représenté au niveau des exemples d'apprentissage.
b) Chaque entrée e1 est la moyenne arithmétique de deux autres valeurs tirées aléatoirement de façon uniforme. Les entrées e2 sont uniformément réparties (montrées sur la figure)

On peut construire en simulation logicielle des cartes 3D (6 voisins), voir 4D ou nD.

Les cartes auto-organisatrices trouvent une certaine justification biologique dans l'existence au niveau des cortex moteur et sensoriel de cartes somatotopiques. Chaque partie du corps est représentée : c'est l'homonculus (fig. 9).

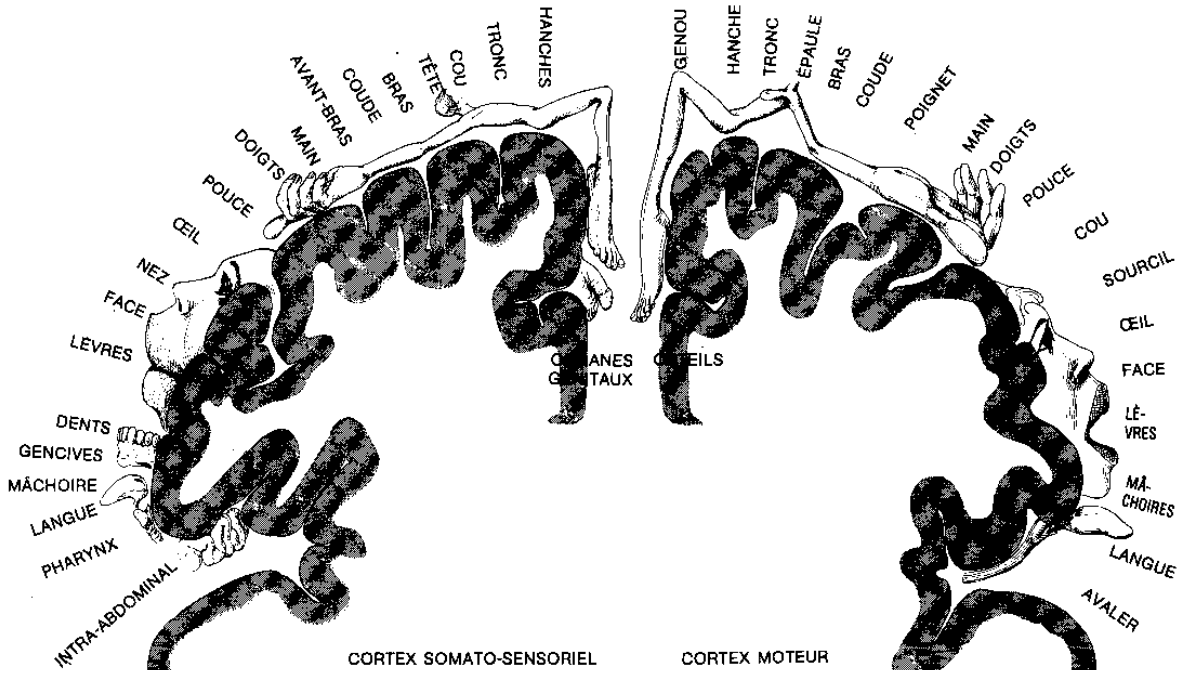


Figure 9. L'homonculus : la surface occupée au niveau corticale est fonction de la sensibilité sensorielle ou précision motrice de la partie du corps correspondante. Ainsi, la surface occupée par le pouce est supérieure à celle de la cuisse (arrangement spatial optimal). D'autre part, la topologie est conservée : les doigts sont l'un à coté de l'autre, etc.

Remarque : l'amélioration des performances de l'algorithme d'apprentissage peut être obtenue par l'introduction de deux nouveaux paramètres d'action comparable au phénomène biologique d'accoutumance (cf annexe).

5 Application à la robotique

On utilise la capacité de représentation spatiale de la carte auto-organisatrice pour piloter un bras de robot. Il s'agit de mettre en correspondance l'espace cartésien dans lequel travaille l'opérateur humain avec l'espace du robot (coordonnées de chacun de ses axes en valeurs angulaires). Lorsque cette transformation de coordonnées est réalisée de manière algorithmique, elle fait appel à des inversions de matrices, coûteuses en temps de calcul, qui génèrent éventuellement des problèmes de conflit lorsque la solution n'est pas unique, ou lorsque les limites de la mécanique (butée) interdisent certaines positions. Le passage coordonnées cartésiennes / coordonnées angulaires et réciproquement est très simple en utilisant la carte auto-organisatrice.

Structure : une carte auto-organisatrice de n x n neurones (habituellement quelques centaines) avec 6 entrées par neurones (3 coordonnées cartésiennes et 3 coordonnées angulaires) réalise la

partie sensorielle du système. La partie motrice est confiée à une couche de 3 Perceptrons (recevant une entrée de chaque neurone de la carte), dont les sorties pilotent les moteurs du robot (fig. 10).

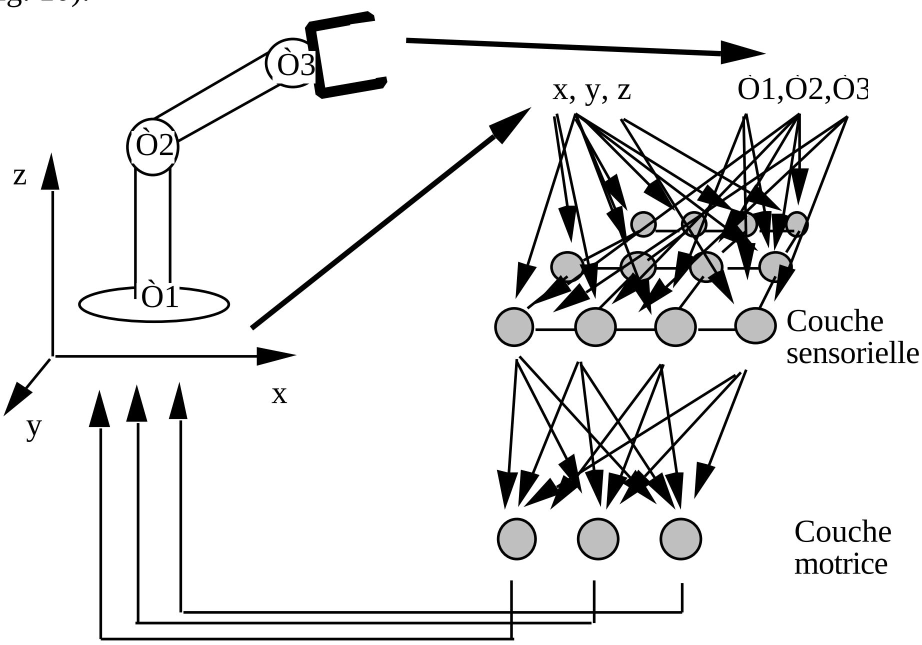


Figure 10. Architecture de Neurobot (Y. Coiton). Chaque neurone de la carte auto-organisatrice reçoit 3 entrées cartésiennes (espace de l'opérateur) et 3 entrées angulaires (espace du robot). La couche motrice se compose de 3 Perceptrons correspondant aux 3 valeurs angulaires pour chacun des 3 moteurs du robot.

Fonctionnement : une position pour l'extrémité du bras de robot est spécifiée en coordonnées cartésiennes par l'opérateur. On récupère en sortie de la couche motrice les coordonnées angulaires correspondant à la même position. En fait, si cette postion à atteindre est éloignée, un certain nombre de positions intermédiaires sont générées par le réseau (fig. 11).

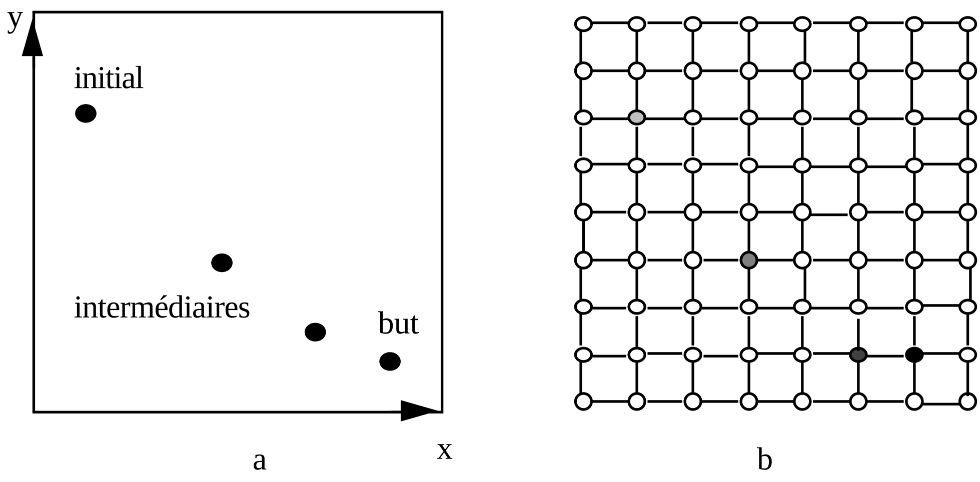


Figure 11. Illustration du fonctionnement dans le plan (x, y).

a) Lorsqu'un point à atteindre est spécifié en coordonnées cartésiennes, le système génère un certain nombre de positions angulaires intermédiaires qui conduisent au but.
b) Au niveau de la carte auto-organisatrice, tout ce passe comme si on avait deux foyers d'activité (initial et but). Le principe de fonctionnement de la carte impose de réduire les foyers d'activité à celui du but, générant par là des foyers intermédiaires.

Apprentissage : la base d'apprentissage est construite en temps réel. Le bras de robot est déplacé aléatoirement dans l'espace des configurations possibles. Pour chaque position, les capteurs appropriés fournissent à la carte les 3 coordonnées cartésiennes et les 3 coordonnées angulaires. L'aprentissage sur la couche motrice est réalisé par la règle du Perceptron à partir des coordonnées angulaires correspondantes. La durée de l'apprentissage est importante (quelques heures) du fait de la lenteur de déplacement de la mécanique. Quelques milliers d'exemples de positions sont nécessaires.

Remarque : la précision du déplacement est fonction du nombre de neurones sur la couche sensorielle. Plus ce nombre est élevé, meilleur est la précision. D'autre part, vu que l'espace à modéliser est à 3 dimensions (cartésien), les réseaux auto-organisateurs 3-D (6 voisins) sont plus performants.

6 TP Compression d'images par carte auto-organisatrice

La quantification vectorielle est l'une des méthodes de compression d'image parmi les plus employées. Nous allons réaliser cette quantification en utilisant des cartes auto-organisatrices.

1/ Introduction

Les images qui proviennent de la télévision, de visioconférence, de satellite, ou d'applications médicales … quelles qu'elles soient, représentent une quantité énorme de données après digitalisation. Diminuer les coûts de transmission ou de stockage est d'un intérêt très ancien (par exemple, le code Morse). En ce qui concerne les applications aux communications, le but recherché est alors de minimiser la durée de transmission d'un certain volume d'informations. Ceci apporte à la fois une économie sur le coût, une diminution des risques d'erreurs, une plus grande ergonomie et une plus grande performance, puisque les données sont acheminées en un temps plus court. Les diverses méthodes de compression sont basées sur les techniques de prédiction, de transformation, ou de quantification vectorielle, avec des possibilités de combinaisons entre elles. Elles sont de deux types. Si la transformation réalisée est réversible, la réduction utilise la redondance d'informations et aucune information n'est perdue. Dans l'autre cas, la transformation est irréversible. C'est une réduction d'entropie et il y a perte d'informations.

Les performances de la compression sont mesurées par :

MSE (Mean Square Error) qui représente la différence entre l'image initiale et l'image reconstituée : $\sum$ erreurs^2 ou bien, $\sum$ erreurs^2 / pixel.

Le taux de transmission (Bit Rate)= Nombre de bits / pixel (bpp)
= (log$_2$ nbre de vecteurs / nbre de pixels par bloc).

2/ La quantification vectorielle

Une image est un objet analogique. Sa représentation spatiale est continue de même que la valeur de la couleur en chacun de ses points. Pour modéliser une image, il faut transformer l'image analogique par une fonction d'échantillonnage S, en une fonction discrète f(x,y) (x et y étant les coordonées des pixels) dont l'amplitude représente l'intensité lumineuse. Ce processus est décrit par la figure 12. Dans le cas particulier des images n'ayant qu'une seule couleur, l'intensité en chaque point est appelée niveau de gris. Le nombre plus ou moins élevé de niveaux de gris spécifie la qualité de l'image (par exemple : 256 niveaux de gris sont un critère de qualité).

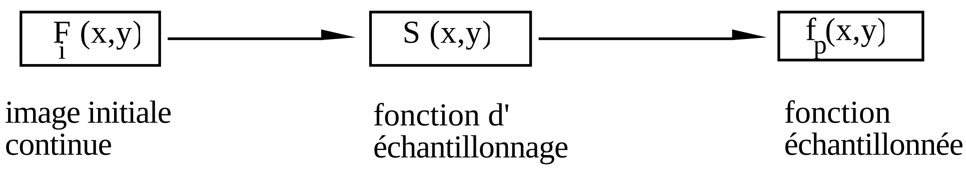


Figure 12. L'échantillonnage

Pour résumer, l'échantillonnage est une numérisation :

- en espace : discrétiser les coordonnées (x,y), c'est échantillonner l'image.
- en fréquence : discrétiser la couleur d'un point, c'est quantifier les niveaux de gris.

Lors de la quantification vectorielle, l'image à transmettre est découpée en blocs de pixels. Chaque pixel peut être codé par les poids des composantes RVB (Rouge, Vert, Bleu) ou par une composante de luminance et deux composantes différentielles de couleur (Y, Cb, Cr). La technique de codage est basée sur le fait, qu'au sein d'une même image de nombreux blocs sont identiques ou peu différents.

La quantification vectorielle réalise une mise en correspondance d'un ensemble de vecteurs d'entrée avec un ensemble de vecteurs de sortie, selon une mesure de distortion. Tous les échantillons d'une région (en fait les blocs semblables) sont représentés par un seul code. Ce code est un index du dictionnaire. Le dictionnaire est composé des blocs les plus fréquents de l'image, ce sont les "mots" du dictionnaire. Sur la ligne de transmission, les codes correspondant à chaque bloc de l'image initiale sont envoyés séquentiellement. A la réception,

on utilise le dictionnaire pour reconstruire l'image, qui est donc établie seulement à partir des mots du dictionnaire.

Le principal problème de la quantification vectorielle réalisée par l'algorithme classique (Linde-Buzo-Gray ou LBG) est la non-invariance du dictionnaire par rapport aux conditions initiales. En effet, on est obligé de fixer des valeurs initiales pour les mots du dictionnaire, valeurs qui influencent la solution finale. De fait, cette solution n'est pas toujours optimale. D'autre part, pour construire un dictionnaire optimal, on utilise une mesure de distorsion entre l'ensemble d'entraînement et le dictionnaire. L'algorithme est sous-optimal puisqu'il tend à surcoder les régions denses de points alors que les régions clairsemées sont sous-codées. Les blocs peu représentés (et éventuellement très significatifs pour l'oeil) sont rejetés au profit de blocs plus courants (codant le fond par exemple).

3/ Principe de la quantification vectorielle par carte auto-organisatrice (fig. 13)

Le nombre de mots du dictionnaire est égal à la taille du réseau (nombre de neurones). Le dictionnaire est donc composé des blocs les plus représentatifs de l'image.

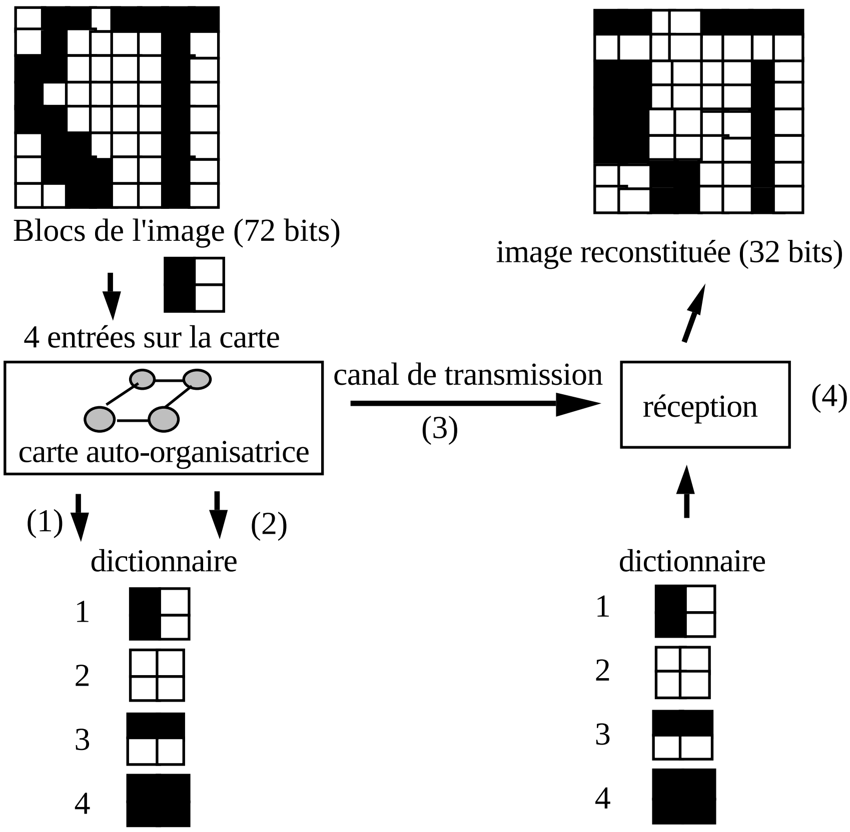


Figure 13. Principe de la quantification vectorielle par carte auto-organisatrice

(1) Construction du dictionnaire : la carte sélectionne les mots du dictionnaire à partir des blocs les plus représentatifs de l'image.

(2) Codage de l'image (par le dictionnaire) : la carte sélectionne pour le bloc de l'image qui lui est présenté le numéro du mot du dictionnaire le plus proche.

(3) Transmission : le numéro des vecteurs est transmis par le canal.

(4) Décodage (reconstitution).

Questions : Etudiez les différentes possibilités offertes par l'utilisation de la carte auto-organisatrice pour la quantification vectorielle d'images. Comparez les performances obtenues pour les différents paramètres suivants :

1/ Dictionnaires de tailles différentes (liées à la taille de la carte, c'est-à-dire le nombre de neurones qui coderont les blocs),

2/ Cartes de dimensions différentes (l'espace sera projeté sur un réseau à 1 dimension, 2 dimensions et 3 dimensions),

3/ Taille des blocs , c'est-à-dire la dimension des vecteurs d'entrée,

4/ Longueur de la séquence d'apprentissage (nombre d'itérations). On cherchera à optimiser la durée d'apprentissage),

5/ Influence sur le voisinage (par µ et ß qui sont les gains de modication de poids).

Réponses : A titre indicatif, nos expérimentations utilisent une image de 172 x 128 (22 016 pixels) sur 8 niveaux de gris.

1/ Variation de la taille de la carte

La dimension des vecteurs est de 2 x 2 = 4 pixels. Il y a 20 itérations d'apprentissage (µ= 0.8 et ß= 0.7).

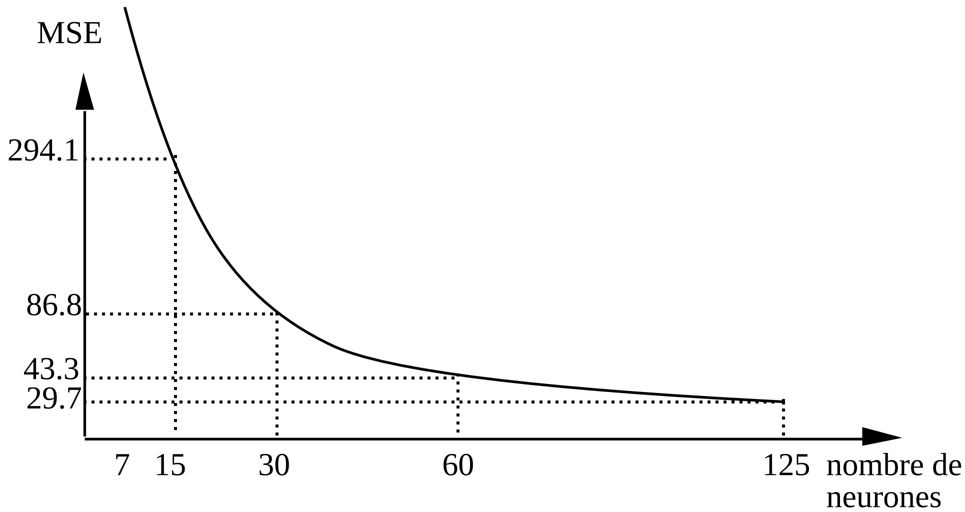


Fig. 14 Variation de la taille de la carte auto-organisatrice

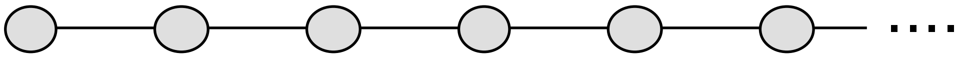
Plus le nombre de neurones est important, meilleure est l'image restituée ; mais le taux de compression varie à l'inverse. Il faut trouver un compromis entre la qualité de l'image et le taux

de compression. Nous avons choisi pour le suite de l'expérimentation 30 neurones (6 x 5) et des blocs de 2 x 2.

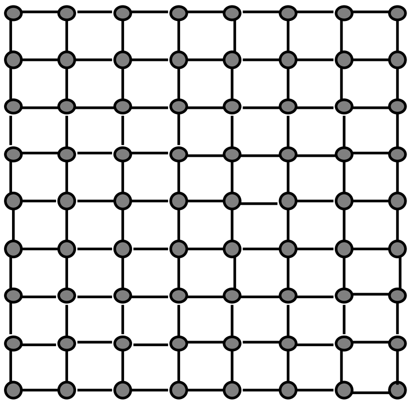
2/ Variation des dimensions du réseau : réseau 1D, 2D, 3D

Nous choisissons d'effectuer seulement 10 itérations et les valeurs suivantes : $\mu = 0.5$ et ß = 0.1, les blocs sont de 2 x 2. La dimension de l'espace d'entrée est égale à la taille des blocs. L'hypothèse que nous voulons tester est : la projection faite sur l'espace de ce réseau est-elle d'autant meilleure que les dimensions du réseau et des entrées sont plus proches.

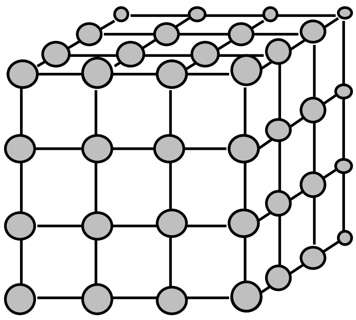
- Réseau à 1D : ligne de 64 neurones



- Réseau à 2D : carte de 8 x 8 = 64 neurones



- Réseau à 3D : cube de 4 x 4 x 4 = 64 neurones



Le meilleur résultat est obtenu pour le réseau ligne (1D). Le voisinage est réduit et tend vers 0 (cf variation du voisinage). Notre objectif est de tester la carte auto-organisatrice pour la compression d'image, nous conservons donc le réseau à 2D pour la suite des tests.

3/ Variation de la taille des blocs (dimension des vecteurs)

Carte de 30 neurones (6 x 5), 20 itérations, µ = 0.8 et ß = 0.7. Plus les blocs sont gros, plus le taux de compression est important. Le problème est qu'il existe une taille limite pour ces blocs à déterminer.

Les blocs 4 x 4 sont très grands, mais ils offrent un bon taux de compression (12.8). Les blocs 2 x 2 donnent un meilleur résultat visuel que les blocs 4 x 1 car dans ce cas, les lignes horizontales sont trop visibles bien qu'ayant le même taux de compression et une erreur sensiblement identique.

4/ Variation du nombre d'itérations (fig.15)

Carte de 30 neurones (6 x 5), blocs de 2 x 2 pixels, µ = 0.8 et ß = 0.7. Plus la durée d'apprentissage est réduite, plus une intégration dans un système réel devient envisageable. Recherchez le meilleur compromis nombre itération/performance.

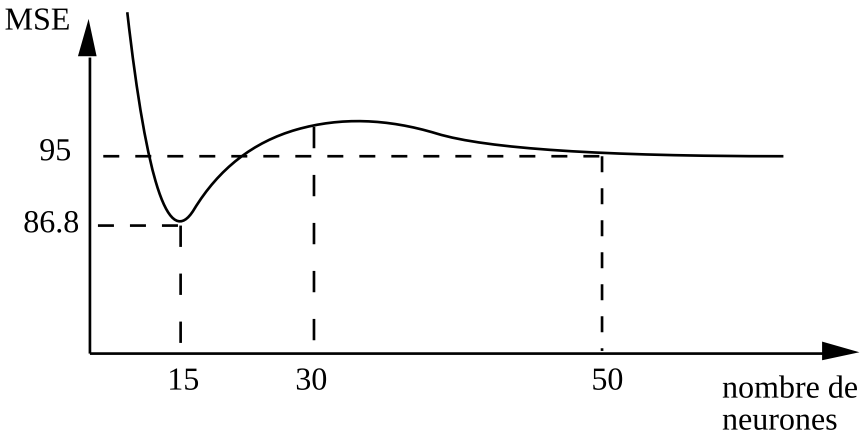


Figure 15. Variation de la séquence d'entraînement

Nous choisissons de nous limiter à 15 itérations pour la suite de nos expérimentations, c'est le nombre pour lequel la distorsion est minimale.

5/ Variation du voisinage

Carte de 30 neurones (6 x 5), blocs de 2 x 2 pixels, 15 itérations. µ est le paramètre de changement de poids du cluster durant l'apprentissage, ß celui de changement de poids pour le voisinage. Recherchez le couple de valeur (µ, ß) qui offre les meilleurs performances. Notez que lorsque µ = 1 et ß = 0, il n'y a pas de voisinage.

Le meilleur résultat (en terme de minimisation de la distorsion) est atteint pour les valeurs (µ = 1 ; ß = 0), c'est-à-dire sans voisinage. Cependant, bien que l'erreur soit supérieure en présence d'un voisinage, à l'oeil l'image semble meilleure. Ce phénomène se reproduit sur plusieurs autres images. Nous constatons ici le fait bien connu que la mesure de la distorsion n'est pas un critère de mesure de la qualité visuelle d'une image compressée.

Résumé des réponses

La variation de la taille de la carte montre que plus celle-ci est de taille importante, plus la qualité de l’image restituée est bonne. Toutefois, plus il y a de neurones, moins le taux de compression est important. Il faut donc rechercher un compromis entre un taux de compression satisfaisant et une bonne qualité d’image.

Concernant la dimension du réseau, plus la dimension est faible, meilleure est la MSE. Cependant, le réseau à 2D est celui que nous avons choisi pour toute la suite de nos travaux, car bien que de qualité absolue inférieure (MSE), il est d’une *qualité visuelle meilleure*. C’est un constat de même type que pour la variation du voisinage.

La variation de la taille des blocs donne un bon résultat pour 2 x 1 pixels (mais le taux de compression serait alors seulement de 1,6), alors qu’à l’opposé les blocs 4 x 4 pixels donnent un mauvais résultat visuel mais la compression est de 12,8.

Le nombre d’itérations de l’apprentissage fait varier l’erreur totale. Pour 15 itérations, on obtient une distortion minimale. Au delà de 15 itérations, c’est le phénomène du “par coeur” qui prend la place de la généralisation.

Enfin, les paramètres µ et ß font eux-aussi varier la distorsion : les meilleurs résultats en valeur de distorsion, sont obtenus pour µ = 1 et ß = 0, c’est-à-dire un réseau sans voisinage. Dans ce cas, on ne peut plus parler de réseau (il n’y a plus de voisinage), le comportement observé est identique à celui de l’algorithme de Linde-Buzo-Gray. Notons cependant que dans ce cas ausi, l'appréciation subjective fournie par la vision humaine ne suit pas toujours les critères quantitatifs.

Enfin, l’utilisation de la carte de Kohonen permet de palier à l’un des problèmes majeurs rencontrés par l’algorithme LBG : la non-invariance par rapport aux conditions initiales. Dans notre cas, le dictionnaire initial est choisi aléatoirement et n’influence pas le résultat final. Rappelons que dans les autres méthodes de quantification vectorielle, ce choix met en oeuvre de nombreux calculs et constitue un facteur déterminant pour la qualité du quantificateur final.

7 Un réseau à architecture évolutive, ART

ART (Adaptive Resonance Theory) est un modèle de réseau de neurones à architecture évolutive développé en 1987 par Carpenter et Grossberg. Dans la plupart des réseaux de neurones, deux étapes sont considérées. La première est la phase d'apprentissage : les poids des connexions sont modifiés selon une règle d'apprentissage La deuxième est la phase d'exécution où les poids ne sont plus modifiés. Avec le réseau ART, ces deux étapes sont réalisées simultanément. Le réseau en phase de test, s'adapte à des entrées inconnues en construisant de nouvelles classes (ajout de neurones) tout en dégradant au minimum les informations déja mémorisées. Il existe plusieurs versions de réseaux (ART1, ART2, ART3). Le réseau ART1 est un réseau à entrées binaires.

1 Structure

Le réseau ART1 est formé d'une couche d'entrée qui est aussi la couche de sortie et d'une couche cachée. Le terme de couche cachée est emprunté au réseau multicouche, il souligne le fait que cette couche n'est pas directement observable par l'utilisateur à la différence de l'entrée ou de la sortie. Il n'y a pas de connexion entre les neurones d'entrées. Par contre, la couche cachée est une couche d'activation compétitive, tous les neurones sont reliés les uns aux autres par des connexions inhibitrices de poids fixes. Chaque neurone de la couche d'entrée est relié à tous les neurones de la couche cachée et, réciproquement, chaque neurone de la couche cachée est relié à tous les neurones de la couche de sortie. A chaque connexion est associé un poids.

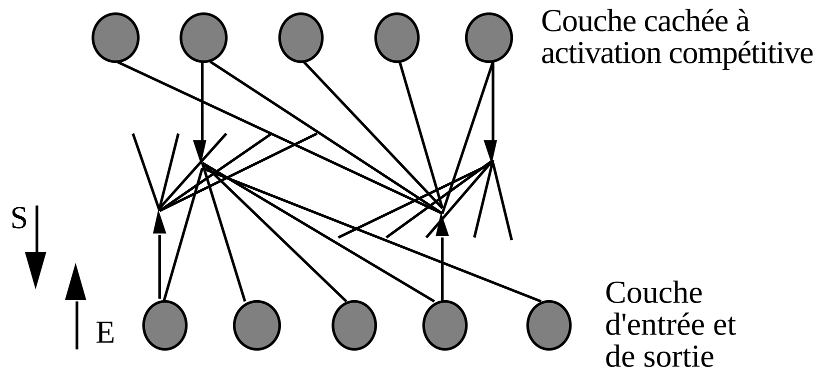


Figure 1. Architecture du réseau ART 1. La couche d'entrée est aussi celle de sortie. Tous les neurones de la couche d'entrée sont reliés à tous les neurones de la couche cachée et tous les neurones de la couche cachée à chacun de ceux de la couche de sortie. Il n'y a pas de relation entre les neurones d'entrée alors que la couche cachée est à activation compétitive.

2 Fonctionnement / Apprentissage

La figure 2 montre un vecteur d'entrée E soumis au réseau. A cette entrée correspond, aprés compétition entre les neurones de la couche cachée, un unique neurone j gagnant. Ce gagnant est considéré par le réseau comme le plus représentatif du vecteur d'entrée E. Le neurone j génère en retour sur la couche de sortie un vecteur S binaire (seuillage). S est ensuite comparé au vecteur d'entrée E. Si la différence est inférieure à un seuil fixé pour le réseau, le neurone gagnant est considéré comme représentant de la classe du vecteur d'entrée. Dans ce cas, la modification des poids des connexions du neurone gagnant a pour effet de consolider ses liens d'activation avec l'entrée E ; en fait l'adéquation entre ce vecteur d'entrée et cette classe est améliorée. Dans le cas contraire, le processus reprend avec les neurones de la couche cachée moins le neurone gagnant de l'étape précédente. Si tous les neurones cachés sont passés en revue sans qu'aucun ne corresponde à E, un nouveau neurone caché est ajouté, qui est initialisé comme représentant de la classe du vecteur d'entrée E.

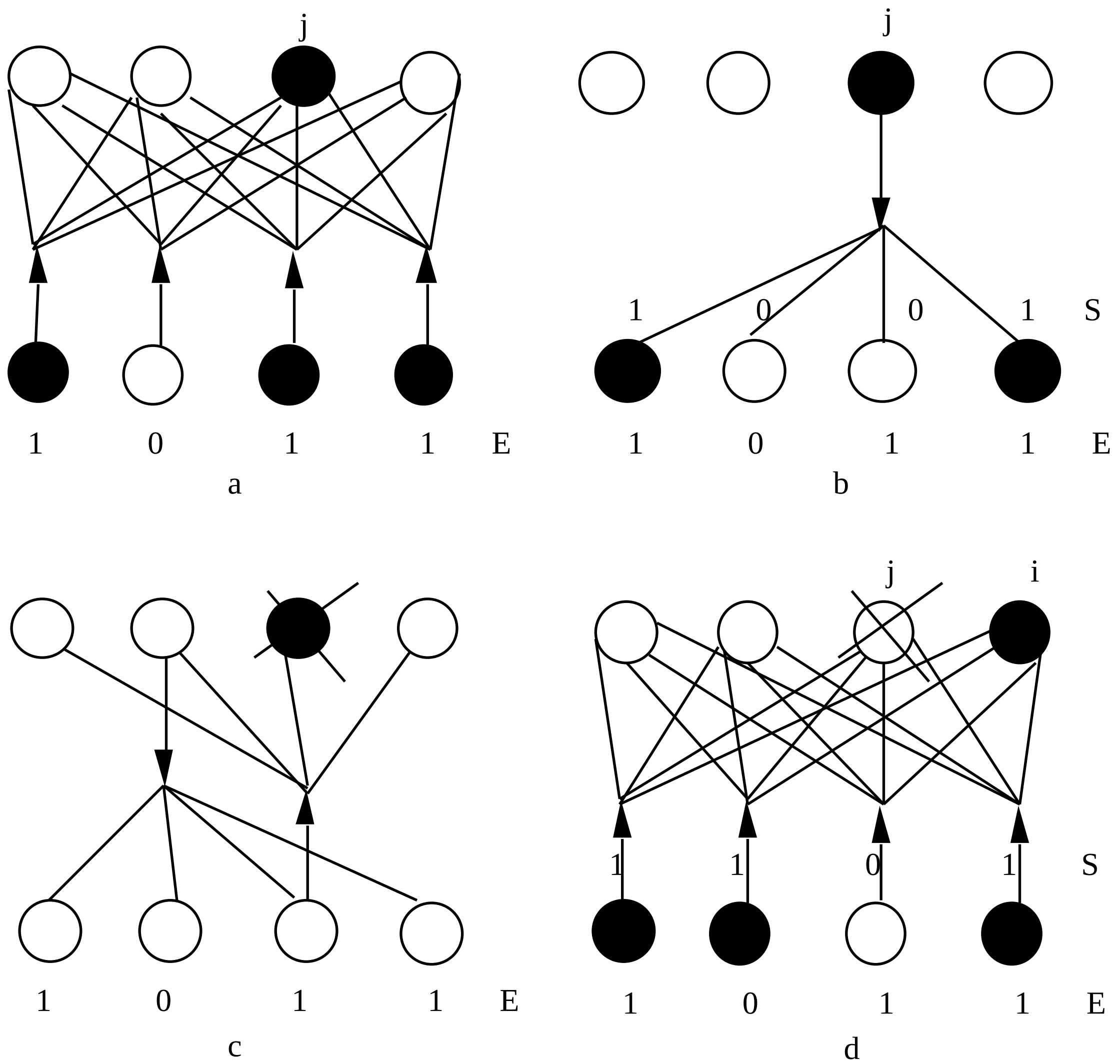


Figure 2. Fonctionnement du réseau ART1.

a) Présentation du vecteur d'entrée E, un neurone gagnant j est sélectionné.

b) Tentative d'unification entre S (retour du neurone j) et E.

c) Echec : suppression du neurone gagnant, présentation de E.

d) Unification : le neurone i est un représentant de la classe du vecteur d'entrée E.

3 Algorithme

Ici, l'apprentissage consiste tout autant dans la détermination des poids que de la valeur du seuil d'unification ß.

1/Initialisation des poids aléatoirement entre 0 et 1 et choix d'un seuil d'unification ß.

2/Présentation d'un vecteur d'entrée E_l appartenant à la base d'apprentissage

3/Calcul du neurone gagnant sur la couche cachée N_j.

4/Génération en retour d'un vecteur de sortie S_j issu de ce seul neurone N_j. S_j a été seuillé afin de le rendre binaire.

5/Tentative d'unification entre S_j et E_l. Soit |S| est la norme de S_j égale au nombre de composantes à 1, par exemple |(1, 0, 1, 1)| = 3.

Si $|S_j| / |E_l| \geq ß$, l'unification est réalisée. Il faut modifier les poids : étape 7.

6/Sinon $|S_j| / |E_l| < ß$, le neurone gagnant N_j est inhibé.

S'il y a encore des neurones non inihibés sur la couche cachée alors retour à l'étape 3.

Sinon un nouveau neurone cachée est créé, initialisé comme représentant de la classe correspondant à la forme d'entrée E_l en utilisant la loi de modification des poids de l'étape 7.

7/Modification des poids

Couche des poids montants :

h neurone de la couche d'entrée, j neurone gagnant de la couche cachée.

$w_{jh} = 1 / |S_j|$ si le neurone h est actif (valeur 1),

$w_{jh} = 0$ sinon (valeur 0).

Couche des poids descendants:

j neurone gagnant de la couche cachée, k neurone de la couche de sortie.

$w_{kj} = 1$ si le neurone k est actif,

$w_{kj} = 0$ sinon.

Retour à l'étape 2.

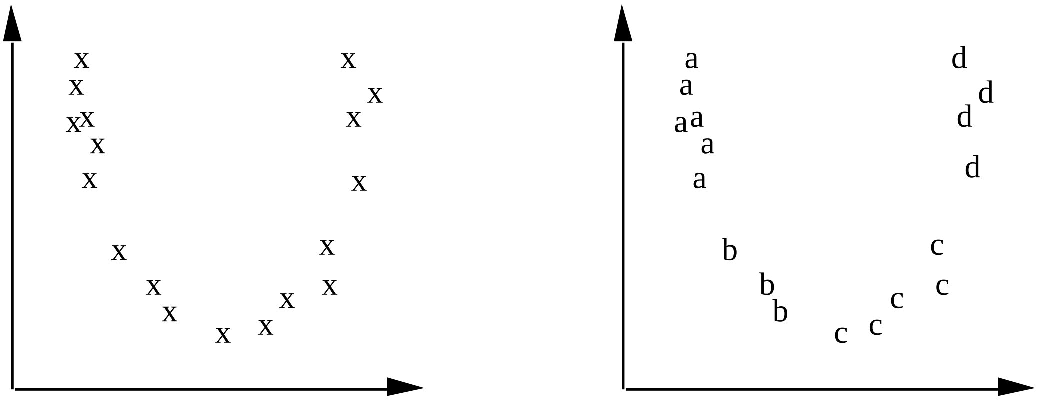
8/Quand le passage de tous les exemples de la base d'apprentissage n'occasionne plus aucun ajout de neurone, il faut mesurer les performances : controler le nombre et la qualité des classes construites. Si le nombre est trop faible, retour à l'étape 1 avec une augmentation de la valeur de ß. Si ce nombre est trop élevé, retour à l'étape 1 en diminuant la valeur de ß.

La valeur du seuil controle le degré d'unification recherché entre les formes à classer et les prototypes des classes. Plus la valeur du seuil est grande, meilleure est l'adéquation recherchée. La valeur du seuil doit être choisie entre 0 et 1. Le neurone i est rattaché à une classe dont le prototype générique à priori ne correspond précisément à aucune des formes de la base

d'apprentissage. L'unification est réalisée lorsque le nombre d'entrées à 1 est comparable avec le nombre de retours à 1 (coactivation statistique).

4 Résultats

Un exemple de coalescence de données issues d'une distribution parabolique est réalisé. Les coordonnées d'un ensemble de point pris sur la parabole sont soumis en données d'entrée au réseau ART1 (fig. 2a). Aprés quelques itérations de l'ensemble de la base d'exemple, les classes construites par le réseau sont présentées sur la figure 2b. Ici quatre classes correspondants aux lettres a, b, c et d sont représentées, la valeur du seuil de vigilance est de 0.7. Plus la valeur de seuil est proche de 1, plus le nombre de classes créées est grand et réciproquement.



a b

Figure 3. Exemple de traitement réalisé par le réseau ART1
a) Base d'apprentissage (points extraits sur une parabole).
b) Coalescence réalisée avec un seuil ß = 0.7 (4 classes).

Les applications majeures du réseau ART ont été réalisé en reconnaissance de la parole, reconnaissance des formes visuelles, en détection d'image radar ainsi qu'en classification et en coalescence.

5 Conclusion

Le réseau ART1 a une architecture en deux couches qui intéragissent entre elles. Le réseau se distingue aussi par deux caractéristiques: sa fléxibillité et sa stabilité pour des entrées arbitraires. Il est capable de s'adapter à des entrées non familières en construisant de nouvelles catégories ou classes (fléxibilité, plasticité) mais aussi d'adapter les classes déja apprises tout en dégradant peu les informations déja mémorisées (stabilité). Le problème posé par ces deux notions antagonistes (fléxibilité-stabilité) est résolu par le principe de l'architecture évolutive.

8 Apprentissage par pénalité / récompense (renforcement)

1 Apprentissage

Cet algorithme d'apprentissage par renforcement est de type qualitatif par opposition aux apprentissages dits supervisé ou non supervisé. Il n'est pas nécessaire de disposer pour les exemples d'apprentissage des sorties désirées, seulement d'une appréciation "globale" du comportement du réseau pour chacun des exemples traités. Cet algorithme s'applique à toutes structures de réseaux. La seule condition est de disposer de neurones de sortie stochastiques (binaires). La réponse du réseau de neurones est ainsi fonction des entrées et, aussi, des neurones de sortie. On introduit donc à ce niveau une part d'aléatoire dans le comportement du système. Si la réponse fournie par le système est considérée comme bonne, l'algorithme tend à favoriser l'apparition de ce comportement en réduisant l'aléatoire. Dans le cas ou la réponse du système globale est considérée comme mauvaise, on cherche à éviter l'apparition ultérieure de ce comportement. Ce processus est itéré jusqu'à l'obtention du comportement désiré pour l'ensemble du système (cf annexe).

2 Algorithme

1/ Les poids sont initialisés à de petites valeurs aléatoires qui placent les probabilités des neurones de sortie autour de 0.5.

2/ Une entrée $E_l = (e_1, \dots e_n)$ est présentée,

3/ Une sortie correspondante possible x_i est calculée pour chaque neurone,

4/ La sortie globale produite est analysée de façon à générer un signal de retour r, positif ou négatif, et une sortie cible (désirée) est choisie :

$d_i = x_i$ si r = +1 (récompense)

$d_i = -x_i$ si r = -1 (pénalité)

5/ La modification des poids est réalisée par la classique méthode du gradient :

$\partial w_{ij} = \mu \cdot r \cdot erreur_i \cdot x_j$

En général, μ dépend de r et est pris 10 à 100 fois plus grand (μ^+) pour r = +1 que pour r = -1 (μ^-).

6/ Tant que la sortie du réseau n'a pas produit une séquence satisfaisante suffisamment longue, retour à 2.

3 Application à l'animation comportementale

A partir des données biologiques sur la connectique nerveuse des insectes, cet algorithme d'apprentissage simule l'apprentissage de la marche. Chez la plupart des insectes, six pattes permettent la locomotion. On ne connait pas complètement les circuits neuronaux impliqués, cependant les travaux histologiques ont montré que chaque patte est doté de son propre

générateur de mouvement et que celui-ci est relié par des connexions intra et intersegments. On postule souvent l'existence d'un générateur central de formes locomotrices. Ce superviseur définirai le comportement de chaque patte. Notons que rien ne soutient cette hypothèse au niveau histologique. Les réseaux de neurones artificiels permettent de montrer qu'un générateur central de formes locomotrices n'est pas nécessaire, son rôle peut être tenu par l'algorithme d'apprentissage. L'espace d'entrée est celui des configurations actuelles de pattes, l'espace de sortie celui des mouvements à réaliser par chaque patte. Les exemples d'apprentissage sont construits au fur et à mesure.

Structure du réseau : Le système se compose des six circuits neuronaux des six pattes, sans superviseur. La structure du système est montrée fig. 1.

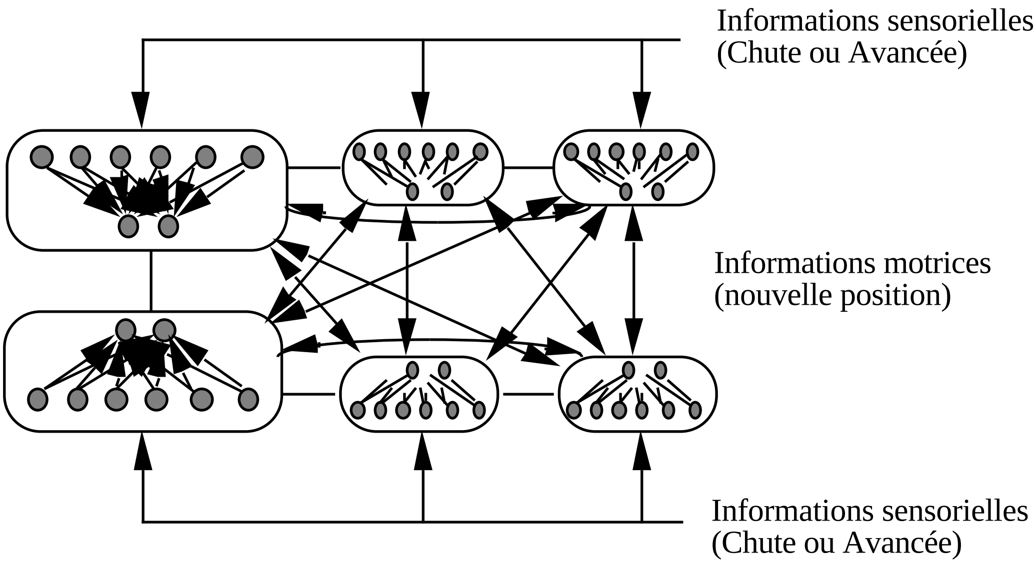


Figure 1. Le système de locomotion est composé de six réseaux neuronaux complètement interconnectés, sans générateur central d'activité. Chaque réseau reçoit des informations sensorielles (chute ou avancée) et des informations sur la position actuelle des pattes qui lui permettent de générer le prochain mouvement de la patte.

Fonctionnement : Les entrées du système sont les informations sensorielles relatives à la position des pattes. Sur la base de ces entrées, le système génère une nouvelle position des pattes. Ainsi, chaque patte a la possibilité de rester inactive, d'avancer ou de reculer. Seule l'action de recul est motrice. L'effet sur l'environnement peut être de trois types : aucun déplacement, avancée ou chute. A partir de cette information, l'apprentissage permet d'éduquer chacun des réseaux pour découvrir une marche. En fait, l'algorithme d'apprentissage explore de manière aléatoire l'espace des mouvements des six pattes. Chaque avancée dans la "bonne" direction est encouragée et on s'éloigne des états conduisants à la chute. Il existe 10 marches possibles à deux temps. A l'issue de l'apprentissage, toutes les marches possibles sont

découvertes (fig.2). Il suffit généralement de moins d'une centaine d'itérations pour trouver une marche.

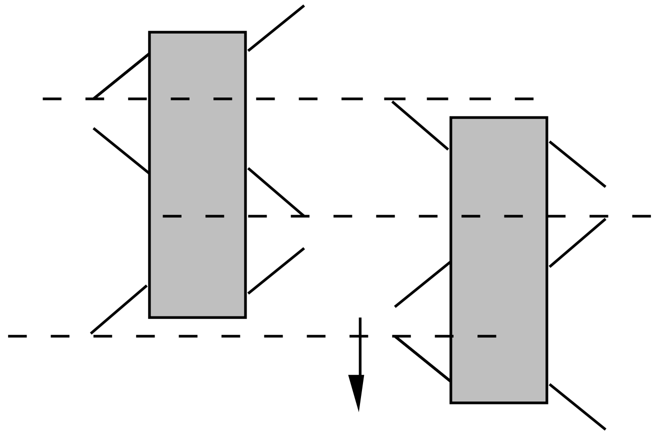


Figure 2. Une marche en deux temps.

9Réseaux multicouches

Apparus en 1985, les réseaux multicouches sont aujourd'hui les modèles les plus employés. Plusieurs couches de traitement leurs permettent de réaliser des associations non linéaires entre l'entrée et la sortie. Ils sont ainsi capables de résoudre le cas du "ou exclusif" (cf Perceptron). On sait depuis les années soixantes que les possibilités de traitement des réseaux multicouches sont supérieures à celle du Perceptron, cependant l'algorithme d'apprentissage manquait. Pour la couche de sortie, on peut appliqué l'apprentissage du Perceptron, mais comment modifier les poids pour les connexions qui ne sont pas en relation avec un neurone de sortie ?

Le problème est ramèné à l'obtention d'une estimation de la valeur désirée pour chaque neurone de la couche cachée. La rétropropagation de gradient est une solution à ce problème. Cet algorithme a été proposé indépendamment par trois équipes en 1985, dont Y. le Cun. Des recherches bibliographiques ont montré qu'il s'agit en fait d'une redécouverte. Concluons que le Faire-savoir est aussi important que le Savoir dans le domaine scientifique.

Le principe utilisé par la rétropropagation ("backpropagation" en anglais) de gradient est la minimisation d'une fonction dépendante de l'erreur. Il s'agit d'une méthode générale, largement employée dans d'autres domaines tels que la physique. Une perception intuitive de cet algorithme consiste à considérer l'apprentissage comme la recherche sur la surface de coût de la position de coût minimal. A chaque configuration de poids correspond un coût. Le gradient est une estimation locale de la pente de la surface. La minimisation du gradient permet de parcourir cette surface orthogonalement aux courbes de niveau d'un pas fixé. Les problèmes rencontrés durant l'apprentissage résultent des zones très plates et des minima locaux.

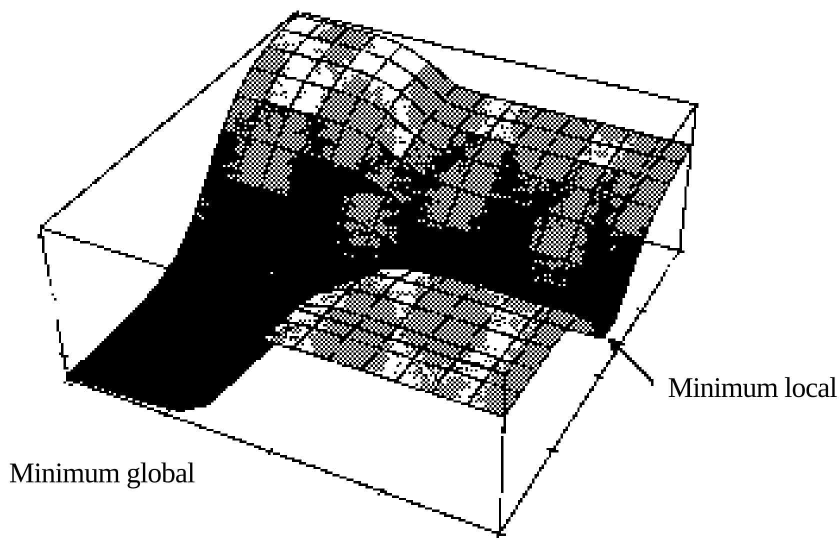


Figure 1. Recherche de la zone de coût minimal dans l'espace de configuration des poids du réseau (ramener ici en deux dimensions) pour les exemples de la base d'apprentissage. Les courbes de niveaux sont en trait fin.

1 Structure / Fontionnement

Les neurones sont continus. La fonction de transfert est une sigmoïde (cf chp 3) qui peut être définie, par exemple, par l'équation :

$f(a_i) = (e^{Òa_i} - 1) / (e^{Òa_i} + 1)$ où Ò spécifie la pente de la sigmoïde.

L'architecture d'un réseau multicouche et son fonctionnement en phase d'utilisation ont été présenté au chp. 3.

2 Apprentissage

L'apprentissage est supervisé : on associe une configuration d'entrée à une configuration de sortie. L'algorithme de la rétropropagation est un algorithme de gradient itératif conçu pour minimiser un critère quadratique ("à la puissance 2") d'erreur entre la sortie obtenue d'un réseau multicouche et la sortie désirée. Cette minimisation est réalisée par une configuration des poids adéquate. L'erreur (e) est la différence entre la valeur désirée (d) pour le neurone de sortie et sa valeur calculée par propagation (x). Ce signal d'erreur permet de définir une fonction de coût :

$$C(W) = M[C_l(W)] = M[\sum_j e_{lj}^2(W)] \quad \text{avec} \quad e_{lj} = (d_{lj} - x_{lj})$$

où, j indique un numéro d'indice pour les neurones de sortie et l indique un exemple d'apprentissage. M est l'opérateur de moyennage, c'est une estimation de la moyenne temporelle dans le cas stochastique. On réalise donc la moyenne des erreurs obtenues pour chacun des exemples de la base d'apprentissage.

Cet algorithme nécessite une fonction continue, non-linéaire et différentiable comme fonction de transfert du neurone.

1/ Initialisation des poids à des valeurs aléatoires de faible grandeur;
2/ Sélection d'un exemple d'apprentissage $(E, d)_l$ dans la base d 'apprentissage
3/ Présentation de la forme d'entrée (E) sur la couche d'entrée du réseau;
4/ Calcul par propagation de la sortie obtenue (o);
5/ Si erreur en sortie alors pour tous les neurones i (depuis la sortie jusqu'à l'entrée)
   Si i est un neurone de sortie alors $y_i = 2 f'(a_i).(d_i - x_i)$;
   Si i est un neurone caché (ou d'entrée) alors $y_i = f'(a_i). \sum_k (w_{ki}.y_k)$;
   (k : neurones compris entre la couche actuelle et la couche de sortie)
6/ Application de la procédure de gradient. µ est un gain fixé par l'utilisateur.
   $w_{ij}(t+1) = w_{ij}(t) + \mu .y_i .x_j$;
7/ Tant que l'erreur est trop importante, retour à l'étape 2 (exemple suivant).

L'algorithme de la rétropropagation de gradient, bien que très simple à implanter, nécessite un certain savoir-faire pour une utilisation efficace. En effet, la convergence de l'algorithme n'est pas prouvée et de multiples variables sont à ajuster précisément en fonction du problème traité. Parmi ces variables à fixer, citons par exemple : les paramètres apparaissants dans les différentes équations (gain de la procédure de gradient (µ), pente de la fonction sigmoïde (Ò), ..), la sélection des exemples pour l'apprentissage et le test, l'ordre de présentation et les distributions relatives des exemples dans la base d'apprentissage, le choix du codage des informations en entrée et en sortie, la structure du réseau (présence éventuelle de connexions directes de la couche d'entrée sur la couche de sortie pour traiter à ce niveau la partie linéaire du problème, limitation pratique du nombre de couches, taille de la couche cachée), la configuration initiale des poids, le nombre d'itérations d'apprentissage, ...

3 Résultats

L'application la plus connue est le logiciel NETtalk (fig. 2). Un réseau de neurones multicouche apprend à prononcer du texte en anglais. Un exemple d'apprentissage est constitué d'une chaine de caractères et de sa transcription phonétique.

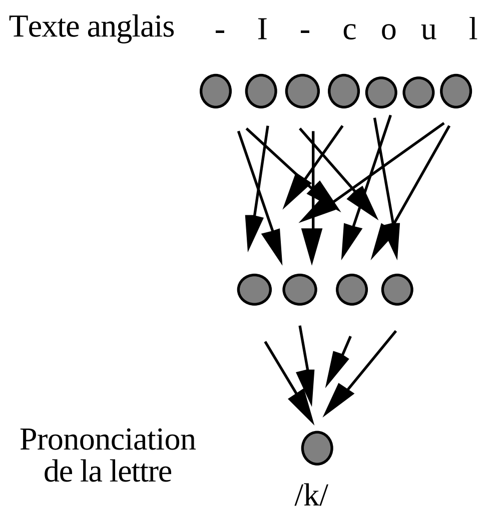


Figure 2. NETtalk : 309 neurones (3 couches) et 10629 poids (80 cellules cachées).
Aprés 12 h d'apprentissage, la performance est de 95% sur le texte d'apprentissage (1000 mots) et 90% sur du texte nouveau (mots nouveaux).
On a connecté en sortie un synthétiseur de paroles, le texte est compréhensible. De plus, il semble que durant l'apprentissage, NETtalk se comporte de la même manière qu'un enfant apprenant à lire (même type de fautes, ...).

4 TP Implication floue calculée par réseau multicouche

1/ Introduction

Nous avons délibérément choisi de coupler connexionnisme et intelligence artificielle, domaines parfois considérés comme concurrents, prouvant ainsi qu'ils ne s'excluent pas. L'expérimentation menée est un exemple typique de système hybride employant au mieux les avantages des deux domaines. L'objectif est de fournir des systèmes experts plus spécifiques du domaine d'application et donc plus performants. Dans les systèmes à règles de production floues actuels, la propagation de coefficients d'incertitude est faite par le truchement de formules de calcul issues de modèles mathématiques du raisonnement incertain. Mais l'utilisation d'une formule fournie par un modèle abstrait général, ne dépendant donc pas intimement de l'application, peut conduire à un traitement de l'incertain ne reflétant pas le raisonnement flou de l'expert humain. L'approche connexionniste proposée résout ce problème de l'inférence floue, la règle de propagation de l'incertitude spécifique du domaine d'application étant déterminée par apprentissage à partir d'exemples d'inférences.

Dans un système à base de règles, la base de connaissances est composée de deux parties : la base de règles et la base de faits. La base de règles contient des connaissances exprimées sous la forme : Si <condition> Alors <action>. Le moteur d'inférences est un programme chargé d'exploiter les règles et les faits. Pour schématiser, on peut dire qu'il consiste à sélectionner, puis à appliquer, les règles dont la partie gauche concorde avec les faits établis. En général, dans ce type de système les règles sont assimilables à des implications logiques entre propositions ou prédicats. Ainsi le raisonnement est lui-même assimilable à une suite de déductions logiques. Dans certaines applications, il est nécessaire de traiter des connaissances incertaines, imprécises ou incomplètes : on parle alors de raisonnement flou (ou, plus précisément, de raisonnement non-exact).

Dans le cadre des systèmes experts, la notion de flou est généralement modélisée par l'introduction des coefficients d'incertitude pris dans l'intervalle réel [0,1]. 0 signifie certainement faux, 1 signifie certainement vrai et les valeurs intermédiaires signifient plus ou moins vrai (et plus ou moins faux). Ces coefficients d'incertitude sont associées à la fois aux faits établis et aux règles. On dispose donc d'une pondération des faits, et d'une pondération de la déduction elle-même.

2/ Inférence floue

La manipulation de règles et de faits flous entraîne plusieurs problèmes pratiques dont l'inférence floue (propagation des coefficients d'incertitude) :

Comment combiner le coefficient d'incertitude d'une règle et celui de sa prémisse pour déterminer le coefficient d'incertitude de sa conclusion ?

Par exemple, soit la règle "S'il y a des nuages alors il va pleuvoir", qui est vrai dans 80% des cas (valeur 0.8). Le temps aujourd'hui est relativement nuageux (à 60%, soit valeur 0.6). Quelle est la probabilité de pluie ?

De manière plus formelle, le problème de la propagation des coefficients d'incertitude (inférence floue) revient à déterminer une fonction g telle que :

CI(Q) = g (CI(P), CI(P->Q)) pour toute règle P->Q du système,

avec :

CI(P->Q) = coefficient d'incertitude de la règle P->Q,
CI(P) = coefficient d'incertitude avec laquelle la condition P a été établie
CI(Q) = coefficient d'incertitude de la conclusion Q, à déterminer.

Dans les systèmes experts actuels, la fonction g est réalisée à l'aide d'une règle de calcul obtenue à partir d'un modèle mathématique d'implication floue (par exemple, l'inférence floue de Lee) ou construite par le cognicien (par exemple, dans le cas de MYCIN). L'inconvénient est qu'il est difficile de garantir que le mécanisme utilisé reflète le raisonnement de l'expert. En effet, il n'existe pas de modèles mathématiques dont on serait certain qu'il traduise parfaitement la problématique du raisonnement flou humain. De plus, un modèle mathématique général est a priori indépendant de l'application et du domaine d'expertise, ce qui ne parait pas être conforme à la réalité. L'analyse des formules utilisées montre avec certitude qu'elles comportent une bonne part d'arbitraire. Cette remarque reste valable en ce qui concerne les formules construites par le cogniticien (cf. par exemple la formule utilisée dans MYCIN).

3/ Inférence floue connexionniste

Nous proposons l'utilisation d'un réseau de neurones pour approximer l'implication floue à partir d'exemples d'inférence floue. Si g est la fonction établisant la correspondance entre les CI des prémisses et celui de la règle avec le CI de la conclusion. Les exemples d'apprentissage à soumettre au réseau seront des vecteurs de la forme : (e1,e2...en,en+1,d) avec d = g (e1,e2...en,en+1). Les valeurs (e1,e2...en) sont celles des CI des prémisses. La valeur en+1 est celle du CI de la règle, d est la valeur du CI de la conclusion. L'apprentissage permet d'adapter la fonction f réalisée par le réseau de neurones au comportement décrit par des exemples d'apprentissage. On peut considérer que le réseau a parfaitement appris lorsque la fonction f est équivalente à la fonction g. La figure 2 montre de façon schématique comment opérer l'apprentissage du réseau.

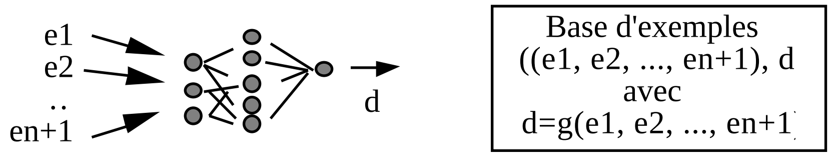


Figure 2. Apprentissage de l'inférence floue par un réseau multicouche,

les exemples sont de la forme (prémisses, conclusion)

4 Inférence floue de Lee

L'inférence floue de Lee est un exemple de modèle mathématique de l'inférence floue. Elle est exprimée par la règle de calcul suivante :

$$CI(Q) = \begin{cases} [0, CI(P\text{->}Q)] & \text{si } CI(P\text{->}Q) = 1 - CI(P) \\ CI(P\text{->}Q) & \text{si } CI(P\text{->}Q) > 1 - CI(P) \\ \varnothing & \text{si } CI(P\text{->}Q) < 1 - CI(P) \end{cases}$$

Pour la réalisation pratique, réduisons la taille du problème en choisissant des valeurs décimales (p = 0.1) pour les CI et une seule prémisse (n = 1) pour les règles.

La table 1 donne le coefficient d'incertitude de la conclusion CI(Q) calculé à partir des coefficients d'incertitude de la condition CI(P) et de la règle CI(P->Q) (pour simpifier le codage on a remplacé l'intervalle [0, CI(P->Q)] par la seule valeur CI(P->Q)).

CI(Q) — CI(P) \ CI(P->Q)	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
0											1
.1										.9	1
.2				Ø					.8	.9	1
.3								.7	.8	.9	1
.4							.6	.7	.8	.9	1
.5						.5	.6	.7	.8	.9	1
.6					.4	.5	.6	.7	.8	.9	1
.7				.3	.4	.5	.6	.7	.8	.9	1
.8			.2	.3	.4	.5	.6	.7	.8	.9	1
.9		.1	.2	.3	.4	.5	.6	.7	.8	.9	1
1	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1

Table 1. Table de l'inférence floue de Lee

Questions : Construction heuristique du réseau

L'utilisation d'un réseau de neurones pour une application particulière ne se fait pas aussi facilement que ce que l'on pourrait le croire. En effet, il n'existe aujourd'hui aucune règle ou formule qui permette de sélectionner au mieux les nombreux paramètres d'un réseau :

1/ Codage des entrées et des sorties,

2/ Architecture du réseau en nombre de neurones et nombre de couches cachées, voir modèle de réseau,

3/ Sélection des bases d'exemples d'apprentissage et de test ...

1/ Codage des coefficients d'incertitude

Les cellules de la couche d'entrée codent le CI de la prémisse et celui de la règle. La sortie est le CI de la conclusion. Il existe de nombreux codages possibles pour les CI sur la couche d'entrée du réseau (fig. 2).

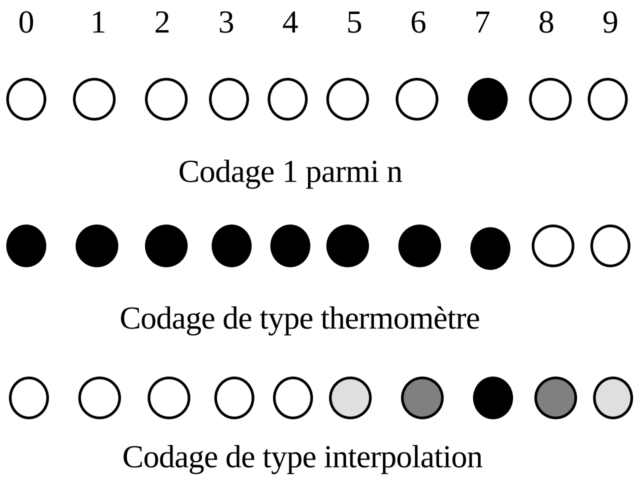


Figure 2. Différents codages possibles pour la valeur 7. Il y a 10 neurone d'entrées, leur valeur d'activation est fonction de la couleur depuis 0 (blanc) à 1 (noir).

Question : Quel est le codage qui vous apparait le plus adapté et pourquoi ? Proposez votre propre codage.

Réponse : Le codage de type thermomètre pour les valeurs de [0, 1]. Ce codage minimise la distance de Hamming entre deux valeurs voisines. Par exemple, le codage de 0.2 et 0.3 ne diffèrent que par la valeur de la troisième cellule en partant de la gauche. On choisi un code éloigné au sens de la distance de Hamming pour ø. Ce codage est montré par la table 2.

CI	Codage
0	1 1 1 1 1 1 1 1 1 1 1 1
.1	1 1 1 1 1 1 1 1 1 1 1 -1
.2	1 1 1 1 1 1 1 1 1 1 -1 -1
.3	1 1 1 1 1 1 1 1 1 -1 -1 -1
.4	1 1 1 1 1 1 1 1 -1 -1 -1 -1
…	
.9	1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
1	1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Ø -1 1 1 1 1 1 1 1 1 1 1

Table 2. Codage des coefficients d'incertitude

Les erreurs réalisées par le réseau dans le cas de l'implication floue de Lee, ne sont pas arbitraires. Elles se produisent principalement dans ce que nous avons dénommé la zone frontière montrée à la table 4.

a \ b, c	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
0											
.1											1
.2											1
.3				Ø						.9	1
.4									.8	.9	1
.5								.7	.8	.9	1
.6							.6	.7	.8	.9	1
.7						.5	.6	.7	.8	.9	1
.8					.4	.5	.6	.7	.8	.9	1
.9				.3	.4	.5	.6	.7	.8	.9	1
1			.2	.3	.4	.5	.6	.7	.8	.9	1

Table 4. Zone frontière de la table de l'inférence floue de Lee.

On remarque qu'il est possible de réduire les erreurs en choisissant les exemples d'apprentissage dans cette zone frontière. Cette zone ne bénéficie pas de l'avantage lié au codage des valeurs des CI, qui a tendance à minorer les erreurs.

2/ Architecture

Question : Le nombre de couches cachées n'est pas fixé a priori. Les résultats théoriques imposent une couche cachée, cependant il est parfois plus pratique d'obtenir une solution avec des réseaux dotés de deux, voir trois couches cachées. D'autre part, si le nombre de neurones des couches d'entrée et de sortie est défini par le codage adopté, il n'en est pas de même du nombre de neurones sur les couches cachées. Recherchez une bonne configuration du réseau.

Réponse : Le réseau sélectionné à la suite d'un processus plus ou moins intuitif associé à une expérimentation exhaustive est composé de 66 neurones, 1080 synapses répartis en trois couches. La couche d'entrée contient 24 neurones, la couche de sortie 12 neurones et la couche cachée 30 neurones.

3/ Bases d'exemples

Le réseau est entrainé à partir d'un ensemble d'exemples qui lui est présenté de manière répétée. Un exemple d'apprentissage est un couple, CI des prémisses et de la règle d'une part, et d'autre part, le CI de la conclusion.

Question : Comment répartissez-vous les 121 exemples entre la base d'apprentissage et la base de test.

Réponse : Les exemples d'apprentissage ont été sélectionnés au hasard parmi les 121 possibles selon une densité de probabilité fixée.

La table 3 montre un ensemble d'apprentissage et son ensemble de test associé.

CI(P) \ CI(P->Q) — CI(Q)	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
0											1
.1										.9	1
.2									.8	.9	1
.3			Ø					.7	.8	.9	1
.4							.6	.7	.8	.9	1
.5						.5	.6	.7	.8	.9	1
.6					.4	.5	.6	.7	.8	.9	1
.7				.3	.4	.5	.6	.7	.8	.9	1
.8			.2	.3	.4	.5	.6	.7	.8	.9	1
.9		.1	.2	.3	.4	.5	.6	.7	.8	.9	1
1	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1

(grisé) : Exemple appris (52), (blanc) : Exemple de test (69)

Table 3. Bases d'exemples d'apprentissage et de test

Question : Dessinez les courbes des performances de l'apprentissage en fonction du nombre d'itérations d'apprentissage et de la taille de la base d'apprentissage.

Réponse : On constate qu'aprés 25 itérations d'aprentissage, un minimum pour la valeur de l'erreur a été obtenu : tous les exemples d'apprentissage produisent la sortie correcte. A l'issu de l'apprentissage, les 81 exemples inconnus de la base de test sont présentés au réseau au cours de la phase de test de façon à mesurer le degré de généralisation. La figure 3 montre la performance de généralisation du réseau en fonction du nombre d'exemples d'apprentissage.

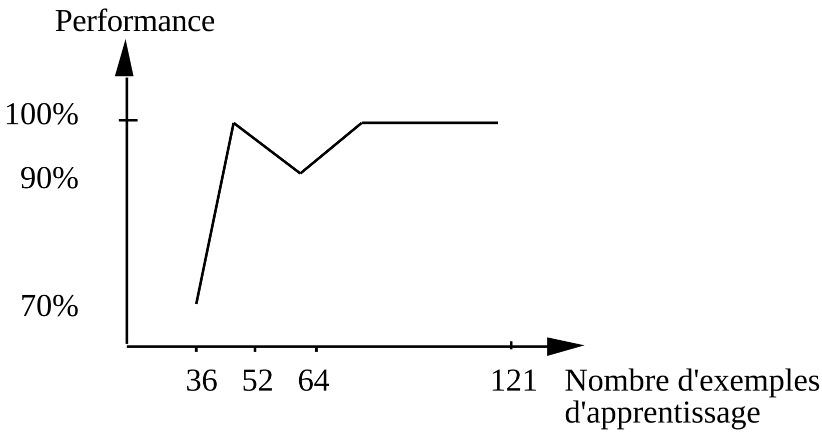


Figure 3. Impact du nombre d'exemples d'apprentissage sur la performance en utilisation

On constate que pour un apprentissage à partir de 40 exemples les performances de la généralisation sont de 72%. De plus, si l'on autorise une erreur d'une valeur absolue égale à 0.1 sur la réponse du réseau, alors les performances de la généralisation par le réseau atteignent 100%.

Performances en fonction du nombre d'itérations : l'ensemble d'apprentissage est constitué de 40 exemples (figure 4) et la base de test est constituée des 81 exemples restants.

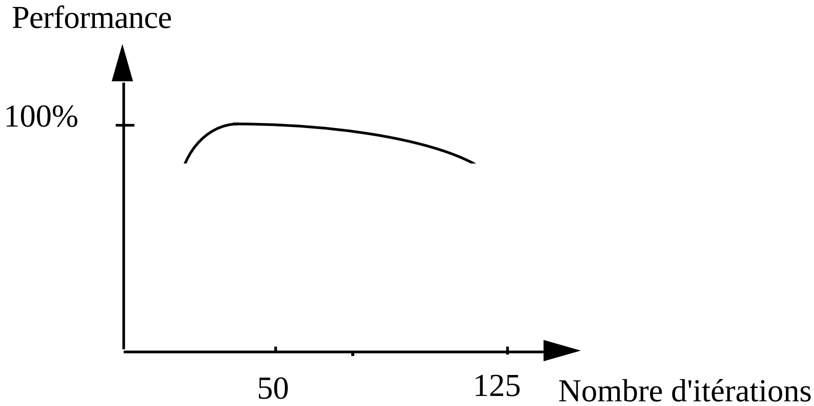


Figure 4. Impact du nombre d'itérations d'apprentissage
sur la performance en utilisation.

On constate qu'après 40 itérations, les performances de la généralisation sont de 100%. Si on considère comme bonne une réponse correspondant à une erreur mesurée nulle, alors les performances sont de 72%.

5/ Inférence floue employée dans MYCIN

Second exemple illustratif des capacités connexionnistes, retrouver à partir d'exemples la formule utilisée dans le célèbre système MYCIN pour la propagation des coefficients d'incertitude. Cette implication floue est décrite comme suit :

Soit $R = F1 \ \& \ F2 \ ... \ \& \ Fn \rightarrow F$ une règle de coefficient CI(R),
on définit alors le coefficient associé à F par : $CI(F) = CI(R) \times v$
avec :

$$v = \begin{cases} w + CIp(F) - w \times CIp(F) & \text{si } w \geq 0 \text{ et } CIp(F) \geq 0 \\ w + CIp(F) + w \times CIp(F) & \text{si } w \leq 0 \text{ et } CIp(F) \leq 0 \\ (w + CIp(F))/1 - MIN(w, CIp(F)) & \text{si } -1 < w \times CIp(F) < 0 \\ 1 \ \text{sinon} & \end{cases}$$

$$w = \begin{cases} MAX(CI(F1), \ ... \ CI(Fn)) & \text{si } CI(Fi) \leq -0.2 \ \text{"} \ i \\ MIN(CI(F1), \ ... \ CI(Fn)) & \text{si } CI(Fi) \geq +0.2 \ \text{"} \ i \\ 0 \ \text{sinon} & \end{cases}$$

$$CIp(F) = \begin{cases} \text{Coeff. précédent de } F & \text{si } F \text{ déjà déduit} \\ 0 & \text{sinon} \end{cases}$$

Faisons observer que dans cette formule :

- le calcul de w cherche à réaliser le MIN en valeurs absolues des CI des prémisses (s'il s'agit de valeurs de même signe et supérieures à un seuil = 0.2)
- le calcul de v cherche à éventuellement "renforcer" le CI d'un fait F provenant de plusieurs règles.
- le coefficient associé à la conclusion est en fin de compte quelque chose comme : CI(R) x MIN (CI(Fi)).

En ce qui concerne l'apprentissage de la formule, nous allons nous limiter (pour des raisons de clarté) au cas suivant :

- Les règles admettent au plus 2 prémisses : $n \leq 2$.
- Les conclusions sont inférées pour la première fois : $CIp(F) = 0$.

Avec ces restrictions, on a :

$$CI(F) = CI(R) \times v$$

$$v = w = \begin{cases} MAX(CI(F1), CI(F2)) & \text{si } CI(F1) \leq -0.2 \text{ et } CI(F2) \leq -0.2 \\ MIN(CI(F1), CI(F2)) & \text{si } CI(F1) \geq +0.2 \text{ et } CI(F2) \geq +0.2 \\ 0 & \text{sinon} \end{cases}$$

Questions : Construction heuristique du réseau

L'utilisation d'un réseau de neurones pour une application particulière ne se fait pas aussi facilement que ce que l'on pourrait le croire. En effet, il n'existe aujourd'hui aucune règle ou formule qui permette de sélectionner au mieux les nombreux paramètres d'un réseau :

1/ Codage des entrées et des sorties,

2/ Architecture du réseau en nombre de neurones et nombre de couches cachées, voir modèle de réseau,

3/ Sélection des bases d'exemples d'apprentissage et de test ...

Réponses

Architecture : le réseau employé comprend 69 neurones et 1100 synapses. Il y a 33 cellules d'entrée, 25 cellules pour la couche cachée et 11 cellules de sortie.

Codage des CI : les CI sont choisies parmi l'une des 11 valeurs de l'ensemble {-1, -0.6, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.6, 1} (la valeur du pas est variable). Le codage sur les couches d'entrée et de sortie reprend celui de la table 3.

Base d'exemple d'apprentissage et de test : il existe 1331 exemples de comportement de la règle d'implication "floue" de Mycin. Comme précédemment, la répartition des exemples entre la base d'apprentissage a été établie de manière aléatoire.

La figure 5 montre les performances de l'apprentissage par rapport au nombre d'exemples appris.

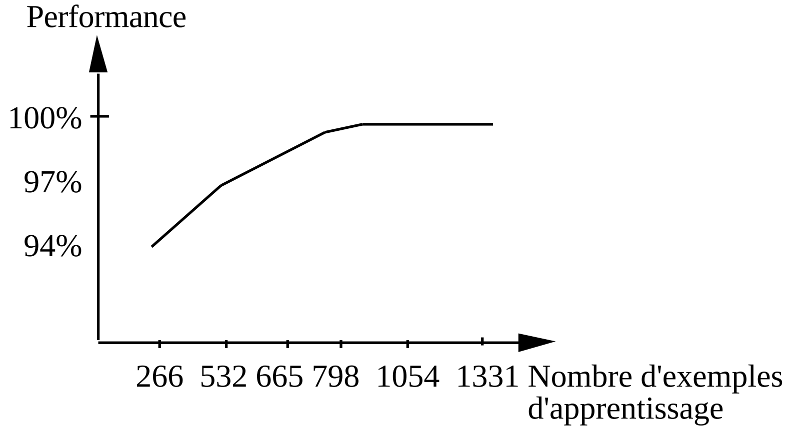


Figure 5. Impact du nombre d'exemples d'apprentissage sur la performance en utilisation dans le cas de l'inférence floue de MYCIN.

665 exemples d'apprentissage sont suffisants pour que les performances du réseau atteignent 100%.

Performances en fonction du nombre d'itérations : l'ensemble d'apprentissage est constitué de tous les exemples (figure 6).

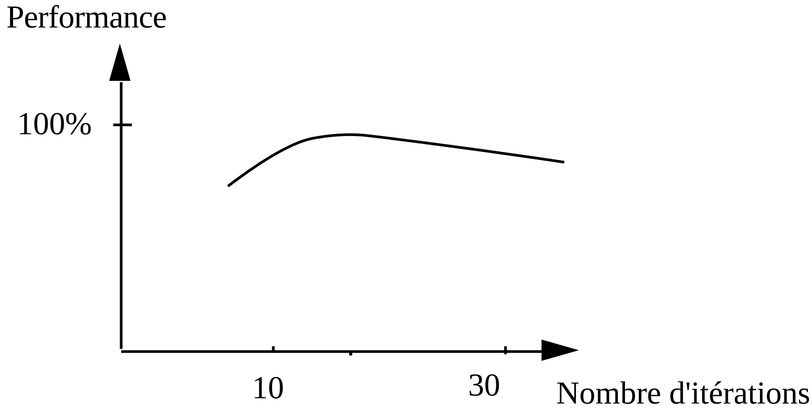


Figure 6. Impact du nombre d'itérations d'apprentissage sur la performance en utilisation pour l'apprentissage de l'implication floue de Mycin.

On constate qu'à partir de 15 itérations d'apprentissage, les performances de la généralisation sont de 100%. Dans le cas d'une mesure à erreur nulle, les performances sont de 84%.

Discussion

Les performances ont été présenté en fonction du nombre d'exemples d'apprentissage. Ceci dans le but d'illustrer les propriétés de généralisation du réseau. Nous sommes à la recherche de la densité d'exemples d'apprentissage minima permettant d'obtenir une bonne généralisation. Cette valeur est évidemment trés dépendante de la nature même de la fonction décrite par les exemples (implication floue de Lee ou de MYCIN).

En fonction du nombre d'itérations, on remarque que la performance en reconnaissance décroit au delà d'une certaine valeur limite. Cela est du à l'apparition du phénomène d'apprentissage par coeur de la base d'apprentissage avec impossibilité de généraliser sur la base de test.

Question : Seul le cas de règle à une ou deux prémisses a été traité ici. 1) Quels sont les problèmes pratiques impliqués par la généralisation aux cas de règles multi-prémisses ? 2) Quelles solutions proposez-vous ? 3) Dans quels cas ?

Réponse :

1) -Le nombre de neurones croit avec le nombre de prémisses.

-La taille de la base d'exemples augmente en fonction du nombre de prémisses et de la précision choisie (nombre de valeurs discrètes disponibles pour coder les CI).

-L'augmention de la précision se traduit aussi par une augmentation du nombre de neurones (à peu prés linéaire).

2) De fait, la mise en oeuvre pratique au sein d'un système réel de notre module connexionniste de propagation de l'incertitude se heurte au problème posé par le nombre, variable, de prémisses des règles. Même en acceptant de fixer a priori une borne supérieure au nombre de prémisses, comment faut-il traiter des règles comportant un nombre variables de prémises ? Une solution coûteuse en nombre de neurones consiste à construire un réseau correspondant à chaque cas.

3) La solution proposée peut être envisagée dans deux cas :
- Si le cogniticien (+ éventuellement l'expert) est capable de trouver par tâtonnements la formule de propagation de l'incertitude (comme on peut le supposer par exemple dans le cas du système MYCIN) : on peut alors envisager de déterminer une formule assez proche par apprentissage et dégager le cogniticien de ce problème.
- Si le cogniticien (+ éventuellement l'expert) est incapable de trouver une telle formule : on peut alors essayer de la construire par apprentissage à partir d'exemples que l'expert sait formuler.

10 Connexionnisme et applications

Les domaines d'application sont ceux de de la reconnaissance de formes, du traitement du signal, du diagnostic, … En fait, est considérée comme éligible toute application qui se représente sous la forme d'une fonction réalisant une mise en correspondance de deux espaces, pour peu que l'on dispose d'exemples représentatifs et en quantité suffisante du comportement de cette fonction. Cependant, déterminer une application potentielle n'est pas tout, il faut aussi spécifier le meilleur modèle de réseau suceptible de résoudre le problème et ses nombreux paramètres.

Remarquons que jusqu'en 1985, date à laquelle la rétropropagation de gradient s'est répandue, les fonctions réalisables par les réseaux étaient limitées aux fonctions linéaires. Cet algorithme d'apprentissage pour réseaux multicouches a permi d'aborder les problèmes non-linéaires (théorème de Hecht-Nielsen en annexe) dont l'exemple le plus représentatif est le ou-exclusif. La Darpa recense un certain nombre d'applications démonstratives réalisées avec ce modèle.

1 Système de mise en correspondance

Les réseaux de neurones sont des systèmes apprenant à réaliser des fonctions de mise en correspondance entre deux espaces.

Par exemple, Y. le Cun a proposé l'application des techniques connexionnistes à un problème de diagnostic médical en cas d'urgence. Dans ce cas, le réseau met en correspondance l'espace de départ constitué des symptômes avec l'espace d'arrivée composé des diagnostics possibles. La fonction associant les symptômes avec les diagnostics est apprise par le réseau à partir d'un ensemble de cas réels (fig.1).

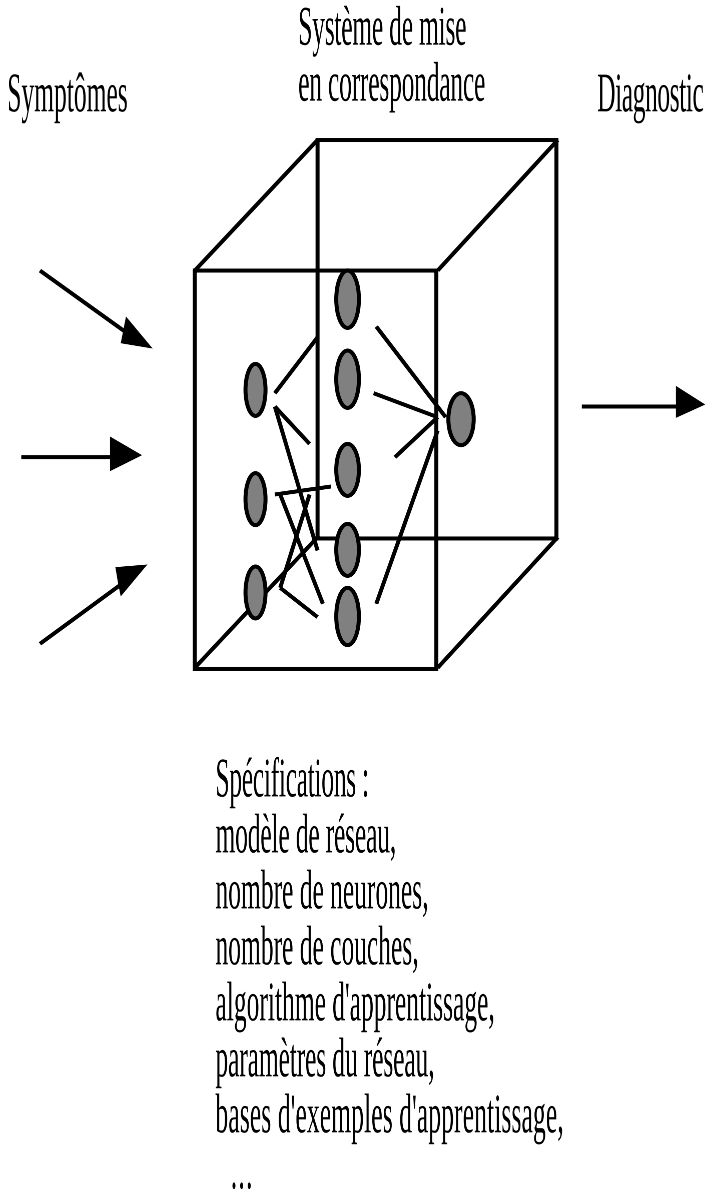


Figure 1. Application à un problème de diagnostic médical

Le terme "application" doit cependant être pris avec précaution. En effet, il faut distinguer entre les "applications candidates" qui sont en principe soluble par la technique connexionniste, les applications en cours de développement dont la faisabilité a été démontré sur un cas simplifié et les "applications prouvées", peu nombreuses.

Réaliser une application, c'est d'abord exprimer le problème sous la forme d'une mise en correspondance de deux espaces, puis construire une base d'apprentissage représentative des données et enfin choisir, en se référant à son expérience, le modèle de réseau et ses paramètres. Il faut aussi préalablement définir les critères de mesure des performances (construction de la base de test), les prétraitements et le codage sur le réseau.

2 Exemple du diagnostic des douleurs abdominales

Expression du problème : Mise en correspondance de l'espace des symptômes avec celui des diagnostics à partir d'exemples de diagnostics réalisés par un expert humain (fig. 1).

Modèle : La base d'exemples est constituée de 5765 malades admis aux urgences. Chaque cas est décrit par 132 symptômes qualitatifs (par exemple le sexe) ou semi-quantitatifs (par exemple l'age), assortis à l'un des 22 diagnostics possibles. Environ 80% des symptômes seulement sont connus pour chacun des malades. On choisi de conserver 1740 cas pour le test (30%) tandis que 4027 cas (70%) sont utilisés pour l'apprentissage. A priori, la relation entre ces deux espaces est complexe (en particulier non linéaire) ce qui implique l'utilisation d'un réseau multicouche (ou de plusieurs réseaux multicouches). En fonction du codage utilisé pour représenter les entrées et les sorties, le réseau se compose d'une seule couche cachée de 62 neurones, de 316 entrées et 22 cellules de sortie. Il y a donc 3380 poids et 400 neurones. Les connexions intercouches ne sont pas complètes.

Performance : Après application de la procédure de rétro-propopagation de gradient, les performances mesurées sur la base d'apprentissage sont de 75% et sur la base de test de 71% (seulement). Il faut relativiser ces performances en ce sens qu'un praticien en situation d'urgence ne fourni le bon diagnostic que dans 60% des cas. D'autre part, l'abscence de certaines valeurs au niveau des symptômes posent problème. De même, il est possible que des incohérences existent au niveau de la base d'apprentissage ou de test (deux situations sypmtômatiques identiques avec un diagnostic différent par exemple). C'est là l'un des principaux avantages de l'approche connexionniste que de pouvoir rester efficace même dans ces situations.

3 Prédiction météorologique (TD)

La prédiction météorologique est un problème très complexe, sans doute à l'origine du développement de la puissance de calcul des gros ordinateurs. On sait aujourd'hui, gràce à la théorie des catastrophes, que les prévisions sont obligatoirement limitées dans leur durée. Après quelques jours, un imprévisible battement d'ailes de papillon au dessus de Pékin (Beijing) modifiera irrémédiablement, et de manière effectivement imprévisible, le temps à Trifouilly-les-Oies. Mais à court terme, les prédictions sont posssibles.

Question : Proposez une approche connexionniste de la prédiction du temps, en explicitant particulièrement l'expression du problème, le modèle utilisé, la nature des données d'entrées, la constitution de la base d'apprentissage et de la base de test.

Réponse : La prédiction du temps peut se concevoir comme une application de diagnostic. Face à une situation non plus composée de symptômes mais plutôt de valeurs barométriques, de

données aérologiques, etc, il faut proposer non pas un diagnostic mais une prédiction. Il y a donc mise en correspondance des données météorologiques actuelles avec le temps à venir à partir d'exemples issus de l'histoire météorologique. Le réseau le plus adapté semble être, dans l'état actuel de la technique, celui qui permet de représenter les relations les plus complexes entre l'espace d'entrée et l'espace de sortie (par exemple non linéaires) : le réseau multicouche. Le codage des données d'entrées est, comme pour le diagnostic médical, primordial et dépend de leurs caractères qualitatif ou quantitatif, ainsi que de leurs domaines de variation.

4 Evaluation de la qualité des plantes en pot

Il s'agit ici d'une application typique du connexionnisme, où l'on demande au réseau d'établir une classification à partir de caractéristiques dont nous ne sommes certains et selon un procesus de combinaison inconnu. En effet, la beauté n'est pas une grandeur quantitative. Elle ne peut pas être mesurée avec certitude (sur quels critères de base ?). Pourtant, la mise sur le marché nécessite la sélection des plantes suivant leur "beauté". C'est une tâche que des experts humains savent remplir.

Expression du problème : Mise en correspondance de l'espace des caractéristiques extraites de l'images avec celui des notations à partir d'exemples de notations réalisées par les experts humains.

Modèle : La base d'exemples est constituée de 100 cyclamens notés par un comité d'experts. Deux images de chaque plante ont été prises, une vue latérale et une vue de dessus. Une segmentation couleur permet de distinguer les fleurs (rouges) des feuilles (vertes). Nous avons alors à notre disposition 4 images par plantes. On extrait de chaque image dix paramètres tels que : la surface totale, le périmètre total, les coordonnées du centre de gravité, les inerties sur les axes horizontal et vertical, et des caratéristiques relatives à l'enveloppe convexe. Il y a donc 40 paramètres pour chaque plantes. Une analyse statistique de l'influence de chacun des paramètres permet de n'en conserver que 8, (a priori les plus importants). Le réseau multicouche choisi se compose de 5 neurones cachés, 8 entrées et 3 sorties correspondants à la note de qualité générale, des fleurs et des feuilles. La base d'apprentissage se compose de 50% des cas.

Performance : Après application de la procédure de rétropropagation de gradient, les performances mesurées sur la base de test montrent que la corrélation du comportement du réseau avec celui de l'expert virtuel est supérieure à 80%. Ces résultats placent l'approche neuronale devant les méthodes purement statistiques, telle que l'analyse en composantes principales.

5 Analyse de données économiques par carte auto-organisatrice

L'objectif de l'analyse de données est de réduire la complexité de l'information afin de dégager des relations de structures, de causalité, en bref l'information pertinente. De nombreuses techniques existent telles que la régression linéaire, l'analyse canonique, l'analyse discriminante, l'analyse en composantes principales (ACP). L'ACP est une méthode de réduction de dimension d'un problème basée sur une composition des caractères (et non une simple sélection). Après cette transformation, il y a projection orthogonale sur le plan principal (sélectionné avec soin pour être le plus discriminant possible). L'application que nous décrivons a pour objectif l'analyse de données économiques de 52 pays en 1984.

Expression du problème : Mise en correspondance de l'espace des données économiques avec l'espace de la carte en respectant la densité de distribution et les relations topologiques.

Modèle : L'état de chacun des 52 pays est spécifiés par 6 caractères, tels que la croissance annuel, la mortalité infantile, le taux d'illétres, la fréquentation scolaire, le PIB (produit intérieur brut) et la croissance du PIB. Il y a donc 52 exemples d'apprentissage. La carte auto-organisatrice est composée de 100 neurones, avec 6 entrées par neurones.

Résultats : Après 35000 tirages aléatoires au sein de la base d'apprentissage, on obtient les résultats montrés figure 2. Chaque pays est positionné sur le réseau d'après le neurone qu'il active. On note l'apparition de groupes de pays correspondant à différentes situations économiques : pays en voie de développement, pays producteurs de pétrole, pays de l'Est et Amérique du Sud, CEE et les 7 pays les plus industrialisés. Notons que l'application de l'ACP sur les mêmes données fournis des résultats comparables (bien que les algorithmes appliqués soient différents).

Suède Suisse	France	Australie	Italie	Yougoslavie Grèce		Koweit	
RFA	USA Japon Canada		Irlande			Bahrein	
Finlande		Espagne		Chili	Mexique	Pérou	Arabie Saoudite
URSS RDA	Royaume Uni		Israël	Venezuela	Brésil		
Cuba		Argentine			Madagascar	Afrique du Sud	Mozambique Nigéria
Corée du Sud Chine	Pologne	Hongrie	Indonésie	Turquie	Maroc Algérie	Sénégal	
Viêtnam	Nicaragua	Syrie Kenya	Egypte Iran	Cameroun Inde	Ethiopie	Niger	BurkinaFaso

Figure 2. Application d'une carte auto-organisatrice aux données économiques :

apparition de groupes de pays.

6 Problème d'optimisation (version connexionniste)

Nous avons vu que les réseaux de neurones artificiels s'appliquent dès lors que l'on peut exprimer le problème comme une fonction de mise en correspondance après apprentissage sur des exemples. Ce n'est cependant pas la seule utilisation qui est faite des réseaux de neurones artificiels. En effet, certains proposent des versions "neuronales" d'algorithmes afin de bénéficier de ce fait d'une exécution en parallèle, d'une implantation VLSI simple ou d'une compréhension plus facile. Citons dans cet ordre d'idées tous les développements réalisés avec le modèle de Hopfield pour des problèmes d'optimisation tels que le problème du voyageur de commerce, ou l'ordonnancement des tâches dans les chaines de fabrication. Il n'est pas dans notre objectif de nous étendre sur cet aspect du connexionnisme. Enfin, et ce sera l'objet des deux applications suivantes, des chercheurs astucieux ont utilisés des "effets de bords" pour résoudre certains types de problèmes particuliers tels que la compression d'image ou le maillage.

7 Compression d’image par réseau multicouche

Principe : L’image à compresser est apprise par le réseau de neurones. Il s’agit d’auto-apprentissage : l’entrée et la sortie sont identiques. La taille de la couche cachée est inférieure à la taille de la couche d’entrée. Après apprentissage, on retrouve l’image à partir des niveaux d’activation des cellules de la couche cachée et des poids des cellules de la couche cachée vers la couche de sortie. On peut coder plusieurs images avec les mêmes poids en ayant des vecteurs d’activation de la couche cachée différents.

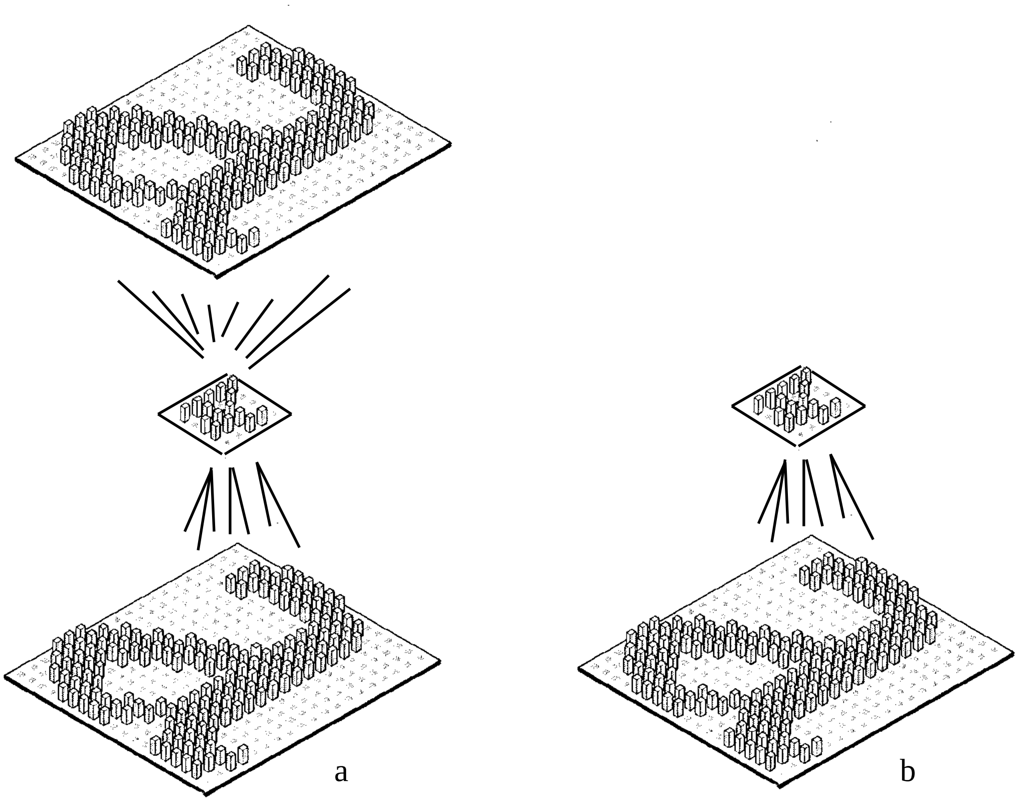


Fig. 8. Réseau multicouche pour la compression d’image.
a) Apprentissage auto-associatif (même entrée et sortie). Le nombre d'unités cachées est, par principe, très réduit.
b) En utilisation, seul le vecteur d'activation de la couche caché est requis, ainsi que les poids des connexions de la couche cachée vers la couche de sortie.

Exemples de taux de compression : Avec un réseau multicouche 100 x 5 x 100 neurones, il y a au total 500 poids (de la couche cachée vers la couche de sortie). Les valeurs des poids sont codées sur 4 bits, ainsi que les 5 valeurs des niveaux d’activation.

- Pour une image initiale de 100 pixels (sur 256 niveaux de gris : 8 bits), on obtient un taux de compression de (100 . 8) / (500 . 4 + 5 . 4) = 0.38

- Si on code 10 images avec le même réseau, on obtient un taux de compression de (100 . 10 . 8) / (500 . 4 + 10 . 5 . 4) = 3.6

Afin que l'opération soit rentable, il faut coder plusieurs images par réseau. Cette technique est limitée par la durée d'apprentissage (plusieurs centaines d'itérations sont nécessaires), ce qui exclu actuellement tout traitement en temps réel.

8 Maillage

Principe : L'idée vous est certainement venue d'interpréter géométriquement la répartition des neurones dans l'espace synaptique. Si cette répartition se superpose à la géométrie d'étude ont peut constituer le maillage d'une pièce. Il ne s'agit pas ici d'une application classique au sens mise en correspondance de deux espaces. Dans de nombreuses disciplines (électrotechnique, mécanique, thermique...), les techniques de résolution numérique font appel à un maillage de la géométrie d'étude. La précision des résultats et le temps de calcul dépendent fortement du maillage qui constitue une étape-clé de la résolution. Les cartes auto-organisatrices du fait de leurs propriétés d'auto-organisation et d'arrangement optimal satisfont aux critères usuels de maillage.

Pour le maillage en électrotechnique, les critères géométriques sont les suivants : aucun " trou " ni recouvrement n'est toléré, les éléments doivent être le plus régulier possible (triangles équilatéraux, quadrilatères proches de rectangles pas trop plats, ...). Les critères physiques sont fonction du problème. En règle générale, le maillage doit être plus fin (éléments plus petits) dans les zones critiques où peuvent se produire des phénomènes tels que : effet de pointe, frontières entre régions, propriétés des matériaux (saturation), ...

Pour mailler une pièce avec une carte auto-organisatrice, la première approche qui vient à l'esprit montre que le réseau ne respecte pas la concavité. Certains neurones sont en dehors de la géométrie. Si on essaye alors de supprimer les neurones en dehors de la pièce et de reprendre l'apprentissage en ayant une forme de réseau plus adaptée à la pièce, les résultats ne sont pas satisfaisants. Il faut réaliser une découpe préalable de la géométrie à mailler en sous-parties convexes et fixer les neurones sur la périphéries (Figure 9).

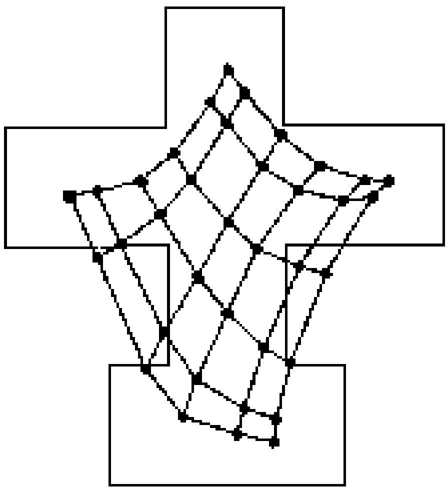


Figure 9. Maillage d'une géométrie concave par une carte auto-organisatrice

Réseau : Le choix du réseau est purement lié à des critères géométriques de la pièce pour la structure et à des critères physiques du problème pour le nombre de neurones (dont va dépendre le nombre de mailles). La forme des mailles est donnée par les relations de voisinage entre neurones. Par exemple, un voisinage de 6 donne des mailles triangulaires.

Résultats expérimentaux (figure 10) : Les résultats présentés montrent la validité d'une telle approche. La qualité géométrique du maillage est satisfaisante. La maîtrise du nombre de mailles est possible. La souplesse au niveau de la non-uniformité du maillage est accrue par rapport aux mailleurs automatiques classiques. La combinaison des éléments (triangles et quadrilatères) est possible. Enfin, cette approche permet une numérotation optimale des nœuds.

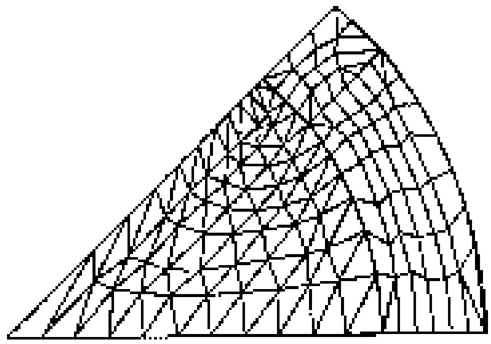


Figure 10 Maillage d'un huitième d'alternateur découpé en trois sous-domaines. L'encoche est maillée par un réseau carré 5 x 5, la partie extérieure du rotor par un réseau rectangulaire 7 x 8. Enfin, le triangle intérieur est maillé par un réseau triangulaire de 12 neurones de côté, les mailles étant triangulaires. Le maillage se compose de 187 éléments. La discrétisation du contour est géométrique aux alentours de l'encoche. On a de plus défini une zone critique maillée plus finement dans le coin supérieur droit du triangle. Le maillage nécessite 500 itérations pour l'encoche, 2000 pour la partie extérieure et 5000 pour le triangle intérieur.

9 Conclusion

Un certain nombre d'applications ont été présenté. Il en existe beaucoup plus, quelques ouvrages référencés dans la bibliographie répertorient uniquement des applications. Dans leur grande majorité, les applications développées sont complètement neuronales. A celà plusieurs raisons, la plus mauvaise est que le développeur ne dispose pas de compétences autres que connexionnistes pour envisager un couplage avec des techniques plus classiques. Une seconde raison est qu'une application purement connexionniste peut être interprétée plus avantageusement en terme de possibilités neuronales. Enfin, et c'est la principale, coupler les techniques classiques avec les techniques connexionnistes est un difficile problème.

11 Développement d’une application en RCM

Il était une fois un ingénieur d'une SSII (Société de Service en Informatique Industrielle) qui avait lu rapidement cet ouvrage. Un industriel se présente alors, son problème est de réduire ses coûts liés à la saisie de fiches manuscrites. Après un tour rapide de la question, il faut se rendre à l'évidence : il n'existe pas dans le domaine de la reconnaissance de caractères manuscrits (RCM) de systèmes fiables, peu coûteux et adaptés à son application. De plus, il n'existe pas non plus d'algorithmes ou de techniques adaptées, au moins dans sa sphère compétence. Par contre, quelques illustrations entrevues dans cet ouvrage rappellent la RCM. L'approche sera donc neuronale.

Le programme de travail a été schématisé sur la figure 1.

1/ Définition du cahier des charges : quelle doit être la vitesse de traitement, quel est le nombre d'écrivains, quelles sont les performances attendues (taux de rejet, taux de mal classés) ?

2/ Construction des bases d'apprentissage et de test. Comment répartir les exemples entre ces deux bases, quel nombre est suffisant, lesquels sont les plus représentatifs ? pour toutes ces questions, les réponses manquent aujourd'hui, aussi on confie souvent au hasard le soin d'y apporter une réponse (qui si elle n'est pas la meilleure, ne devrait pas pour autant être la plus mauvaise).

3/ Sélection d'un modèle de réseau. Il s'agit de réaliser la mise en correspondance de l'image "pixel" avec le caractère. Etant donné que l'on dispose de la sortie désirée, l'apprentissage est supervisé. A priori, la fonction de mise en correspondance est complexe (non linéaire) ce qui impose le choix d'un réseau multicouche. L'algorithme d'apprentissage le plus performant aujourd'hui sur cette structure est la rétropropagation de gradient.

4/ Définition des prétraitements : squelettisation, seuillage, détection de contours, normalisation, ...

Définition du traitement : nombre de réseaux de neurones, association avec traitement classique (vérification, controle de vraissemblance).

Codage des informations d'entrée et de sortie (type thermomètre, pixel, ASCII, ...).

5/ Modèle : un réseau de neurones, scanner (binaire), codage des entrées : on attribue à chaque neurone d'entrée une partie (bloc) de l'image.

Définition : nombre de couches cachées, nombre de neurones par couche (entrée, cachée(s), sortie), constante d'apprentissage (μ), nombre d'itérations d'apprentissage.

Retour à l'étape 4 pour ajustement.

6/ Mise au point sur la base de test

Quelles sont les lettres qui posent problème, pourquoi ?

Sur quels critères peut-on autoriser l'apprentissage en cours d'utilisation, avec quels valeurs de paramètres ?

Vérification du respect des performances précisées dans le cahier des charges.

7/ Essais sur site.

8/ En parallèle, rédaction de la documentation, des drivers pour le scanner, etc.

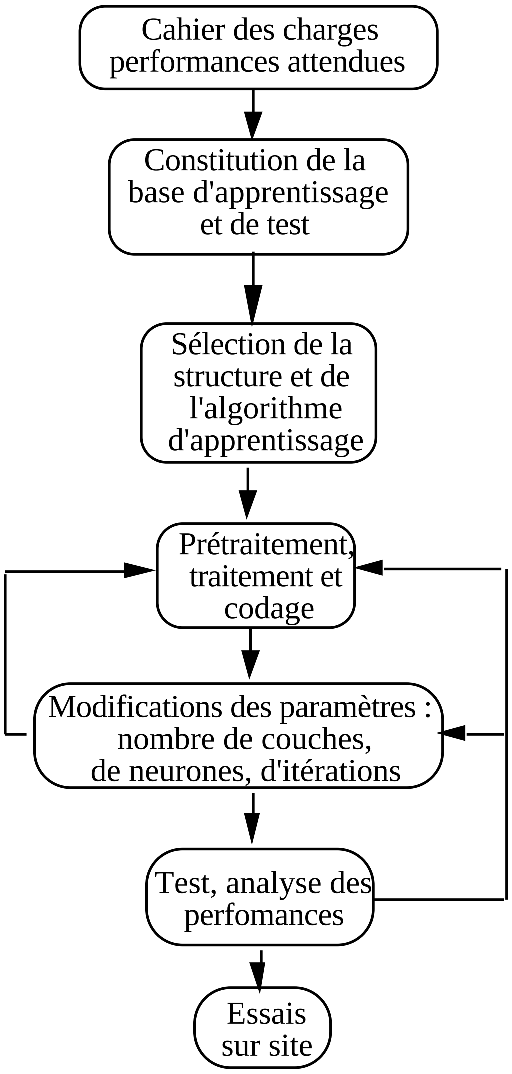


Figure 1. Les étapes de la construction d'une application connexionniste

NB : Dans cette fiction, de nombreux problèmes ardus ont été "oubliés", tels que :

- taille de la base d'apprentissage (si le nombre d'écrivains est important),
- extraction de chacune des lettres de la fiche (et d'une lettre seulement),
- codage de l'image en bloc qui ne fonctionne certainement pas pour toutes les lettres de l'alphabet ensemble. Certaines lettres sont trop peu différentes selon ce codage pour être dissociées.

Le gain principal est, outre celui de proposer une approche à un problème jusqu'alors mal traitée, de réduire le temps de développement. Il faut environ deux mois à un ingénieur non spécialiste des réseaux de neurones pour réaliser un prototype. Cette durée est très inférieure à

celle nécessité par l'étude et le développement des algorithmes employés dans le domaine de la reconnnaissance de caractères manuscrits.

12 Environnements de développement, simulateurs, neurocalculateurs et intégration

Les modèles connexionnistes sont nombreux, en fait plus d'une centaine. Mais leurs spécifications applicatives ne reposent pas sur des résultats théoriques (sauf pour les plus simples : Perceptron). Aussi, l'expérimentation est le moyen le plus approprié pour choisir le modèle de réseau à adopter face à une application déterminée. Aujourd'hui, l'immense majorité des réseaux de neurones artificiels sont simulés logiciellement sur des ordinateurs séquentiels classiques. Nous différencions simulateur et environnement de développement. Un simulateur permet de tester rapidement un modèle que ce soit face à une situation proposée par l'utilisateur ou dans le cadre d'une application prédéfinie. La fonction de ce logiciel est donc a priori informative, en particulier pédagogique. Un environnement de développement a pour objectif de fournir au programmeur un cadre de travail efficace lui permettant de décrire rapidement un applicatif. Il s'agit d'un logiciel beaucoup plus coûteux. En plus d'un simulateur de réseaux neuronaux, il intègre souvent la possibilité de décrire ses propres réseaux et la description des réseaux courants (jusqu'à une vingtaine), des bibliothèques de prétraitement et conversion des données, des outils d'analyse et d'aide à la mise au point, la possibilité de décrire une interface graphique, ... Les fabricants proposent souvent en sus du logiciel des outils pour accélérer les calculs : cartes, processeurs, etc.

Il n'entre pas dans le cadre de cet ouvrage une description des logiciels disponibles sur le marché. Ils sont trop nombreux et d'autre part, leurs fonctionnalités et performances évoluent très rapidement. Notons cependant qu'il en existe pour toutes les bourses : depuis les logiciels en free-ware (on ne paye que le prix du support magnétique), en passant par les versions pour PC, jusqu'aux environnements dédiés pour les stations de travail. Certains industriels proposent même des "neurocomputers". Il s'agit en fait de configurations particulièrement étudiées pour le développement d'applications connexionnistes, qui comprennent souvent des processeurs matriciels, de la RAM, etc.

En ce qui concerne l'intégration de réseaux de neurones artificiels dans des circuits intégrés (VLSI ou autres), plusieurs approches sont explorées par les chercheurs (analogique, digitale, avec ou sans apprentissage). La principale difficulté est liée à l'intégration de l'algorithme d'apprentissage. Dès aujourd'hui, des puces contenant quelques neurones (dizaine) sont en vente, d'autres de quelques dizaines existent à l'état de prototypes dans les laboratoire de recherche.

1 Présentation d'un simulateur

SACREN (Système d'Aide au Choix d'un REseau de Neurones) est un simulateur basé sur les concepts de neurones et de synapses. Il permet la construction et la simulation d'une grande variété de réseaux. Le principal critère guidant son développement a été la flexibilité (chaque cellule possède son comportement propre, tout type de schéma de connexions peut être spécifié), ainsi que le montre la figure 1.

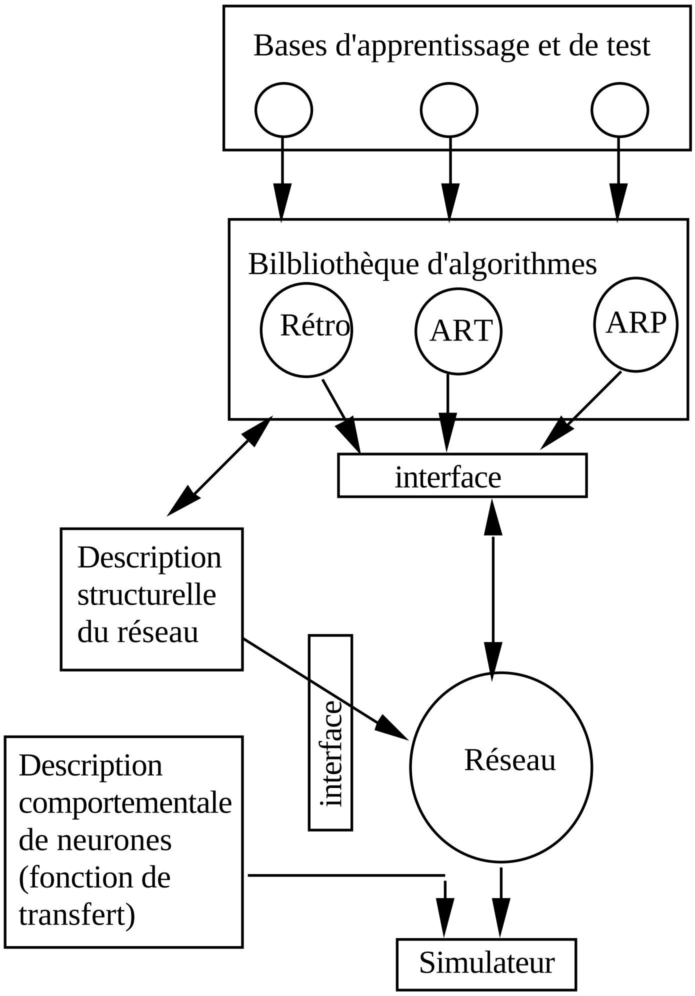


Figure 1. Structure générale du simulateur

Le module désigné sous le terme de description structurelle de neurones regroupe les utilitaires permettant de construire l'architecture du réseau de neurones : les données (schéma des connexions) utilisées pour la simulation. Le second module de description comportementale de neurones a pour objet la spécification du comportement de chacune des cellules. La bibliothèque des algorithmes regroupe différentes procédures d'apprentissage pour le réseau de neurones : la rétropropagation de gradient, etc. La procédure d'apprentissage travaille à partir d'une base d'exemples d'apprentissage. Le module simulateur est composé d'un simulateur évènementiel à évènements discrets qui génère, classe et traite les évènements consécutifs à l'application sur le réseau d'une forme en entrée. Les interfaces sont des procédures proposant à

l'utilisateur, sous forme de menu, toutes les actions impliquées dans le déroulement d'une session.

2 Déroulement d'une session

L'utilisation du simulateur est schématisée figure 2. Le travail débute par le choix de la structure de réseau (modèle, nombre de neurones, comportement de la fonction de transfert) et l'algorithme d'apprentissage (nombre d'itération d'apprentissage, pas de modification des poids, constitution des bases d'exemples). Puis, la phase d'apprentissage sur la base d'exemples est exécutée. L'étape suivante est l'utilisation du réseau qui permet de valider ou d'invalider les différents choix effectués dans les phases précédentes.

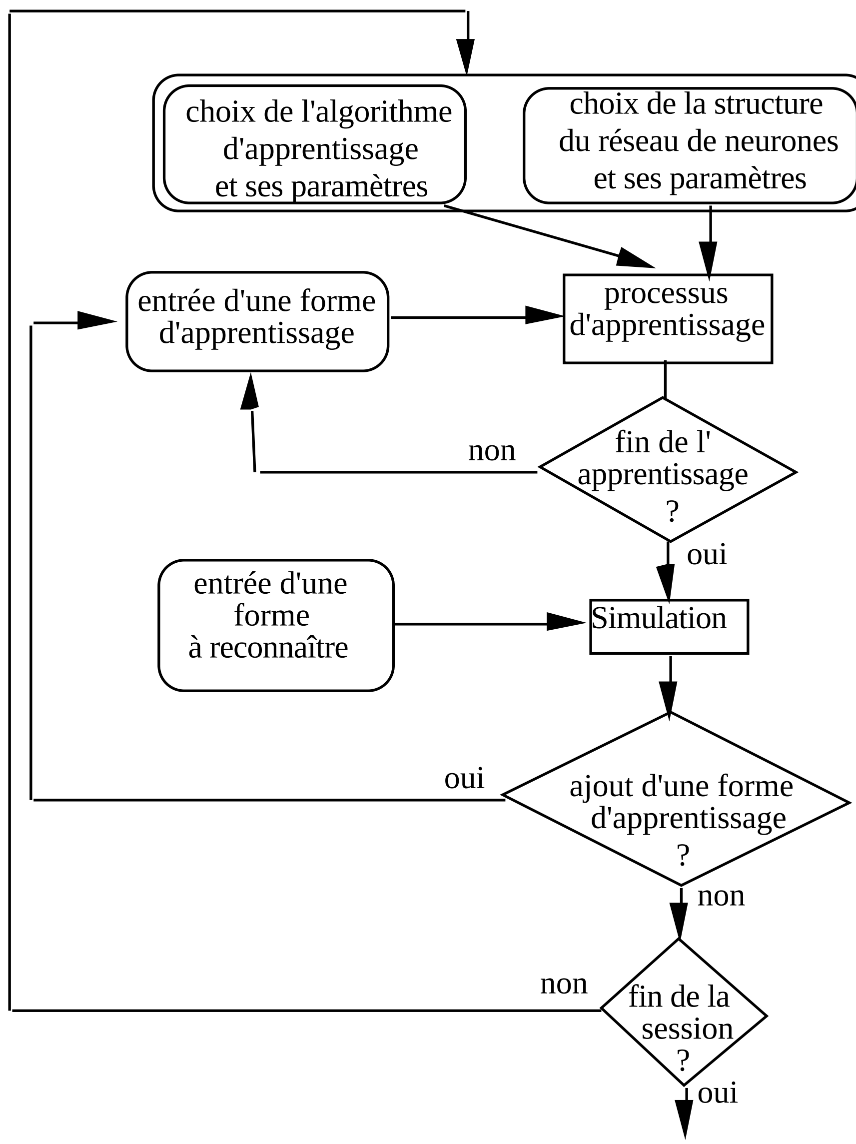


Figure 2. Organigramme d'une session

13 Conclusion

Nous espérons avoir tout au long de cet ouvrage employé un langage clair et concis. Remarquons cependant que si ceci profite à la pédagogie, la rigueur scientifique n'est pas toujours conservée. Il aurait fallu souvent modérer notre propos, annoter d'exceptions tous nos énoncés et consteller le discours de références bibliographiques. Soyons tous conscients qu'il s'agit ici d'une introduction au connexionnisme destinée à donner un aperçu au lecteur curieux et à être complétée par des lectures plus précises par le futur spécialiste. L'approche des réseaux de neurones artificiels s'est faite par référence au monde biologique. Ce n'était pas le seul guide possible. Nous citons au chapitre des renseignements utiles une liste d'ouvrage abordant le sujet selon d'autres démarches (physicienne, mathématicienne, etc).

La description biologique a permi de souligner l'écart entre le modèle et la réalité, par exemple au niveau du nombre d'éléments impliqués ou de leur complexité. Ainsi, le lecteur est à même de revenir sur le terme "réseaux de neurones artificiels" et de se forger sa propre opinion. Bien que très simples, les modélisations réalisées (Perceptron, mémoires associatives cartes auto-organisatrices, ART, ARP, réseaux multicouches) nous ont montrées, au travers d'exemples d'applications, la portée de cette approche pour l'ingénieur. Pour peu qu'il soit capable d'exprimer le problème à résoudre sous la forme d'une fonction de mise en correspondance et qu'il dispose d'exemples pour l'apprentissage, les performances d'une solution connexionniste dans le domaine de la généralisation du comportement à des situations inconnues (ou imprévues) sont tout à fait intéressantes. Citons pour mémoire le diagnostic, la prédiction, la classification, la reconnaissance de formes, etc. Même lorsque les performances ne sont pas supérieures à celles d'une approche plus classique (algorithmique ou à base de connaissance), la facilité de programmation des modèles neuronaux par l'exemple autorise le développement d'applications sans requérir une connaissance très exhautive du domaine par l'ingénieur. Les temps de développement, et les coûts, sont donc réduits. Remarquons cependant que si une expertise du domaine d'application n'est pas nécessaire de la part de l'ingénieur, il lui faut malgré tout acquérir une aisance dans la manipulation des modèles et leurs paramètres, la constitution des bases d'apprentissage, et d'autres facteurs moins identifiés. Celle-ci ne peut actuellement être obtenue qu'à l'issue d'une expérience personelle de la mise en oeuvre de techniques neuronales, dont les travaux pratiques proposés ici peuvent constituer la base.

Il est temps maintenant de questionner la validité de certaines hypothèses fondatrices. Ainsi, nous avons supposé que les comportements intelligents s'appuient sur l'architecture neuronale au niveau des neurones et de leurs connexions. Les nombreux exemples applicatifs que nous avons recencé semblent confirmer ce choix. Cependant, dans quelle mesure n'est-il pas possible d'observer des comportements plus intéressants (plus intelligents ?) en choisissant comme niveau de modélisation les membranes ioniques, les molécules ou les comportements

élémentaires. La figure 1 montre les différents niveaux biologiques et le niveau de modélisation choisi par les réseaux de neurones artificiels.

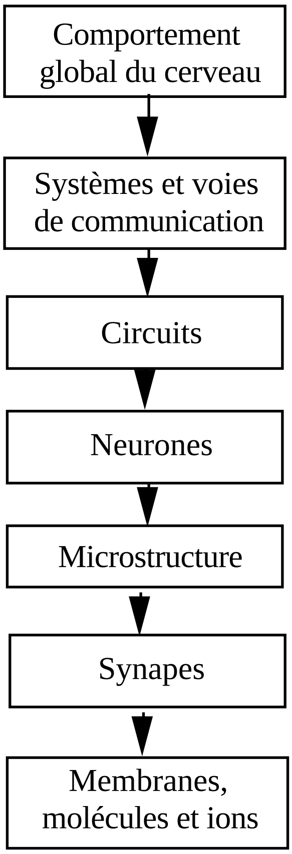


Figure 1. D'autres niveaux de modélisation possibles
pour la génération de comportements intelligents.

Remarquons que le terme "ordinateur moléculaire" est déjà sorti des livres de science-fiction. Une conférence sur le sujet organisée en 1983 à l'initiative de la NSF (National Science Founding, USA) appellée "Chemically Based Computer Design" (construction d'odinateurs basée sur la chimie) a cristallisé et catalysé les efforts de recherche. Notons aussi l'apparition du terme "neuro-éthologie computationelle" où les réseaux de neurones artificiels fournissent un support physique aux comportements élémentaires, dont l'interaction génére un comportement complexe.

Le lecteur curieux aura sans doute remarquer que la conclusion ne semble pas annoncer la fin de cet ouvrage, tout au moins au vu du nombre de pages restantes. En effet, nous proposons quelques questions récapitulatives destinées à tester le lecteur. De plus, nous avons délibéremment choisi de reporter en annexe tout le formalisme mathématique non indispensable. Suivent les références indispensables pour prolonger l'instruction et quelques informations pratiques. Enfin, il nous semble important de proposer un index et un glossaire fournis.

14 Questions récapitulatives

1 Association d'une carte auto-organisatrice avec un réseau multicouche

Question : Si l'on place en prétraitement d'un réseau multicouche à rétropropagation de gradient une carte auto-organisatrice, va-t-on augmenter les performances par rapport à un réseau multicouche seul ? Argumentez.

Réponse : Non, car la sortie de la carte est un unique foyer d'activation.

2 Machine séquentielle connexionniste

Il existe un certain nombre d'applications où la réponse doit être fonction des entrées actuelle et passées. A l'exemple du domaine de la reconnaissance de la parole, le traitement réalisé doit pouvoir prendre en compte la séquence des événements reçus et non plus seulement l'entrée actuelle (fig. 1.). Nous définissons une séquence comme une suite ordonnée de formes. Une forme est un vecteur d'entrée. La longueur, ou taille, de la séquence est égale au nombre de formes qui la composent. Dans une séquence, la même forme d'entrée peut produire des sorties différentes.

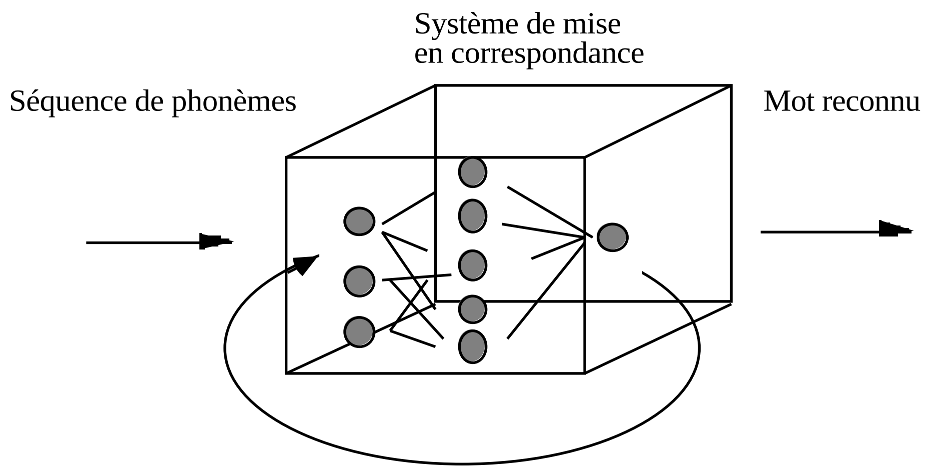


Figure 1. Application à un problème de nature séquentielle

Un certain nombre de modèles connexionnistes ont été proposés pour résoudre des problèmes de nature séquentielle. Mise à part deux approches qui évacuent le problème séquentiel en le transformant en un problème combinatoire (série-parallèle et spatio-temporelle), les autres approches sont toutes des instances du modèle général de machine séquentielle (fig. 2).

La machine séquentielle est une représentation formelle proposée pour la modélisation des circuits séquentiels, qui englobe tous les modèles de réseaux séquentiels. Une machine séquentielle est un quintuplet

$$M = (I, O, S, \delta, \lambda)$$

où I, O, S sont respectivement les ensembles non vides, finis, des entrées, des sorties et des états.

$\delta : I \times S \rightarrow S$ est la fonction de transition. Elle calcule l'état interne futur en fonction des entrées actuelles et de l'état interne présent.

$\lambda : I \times S \rightarrow O$ est la fonction de sortie. Elle évalue la sortie en fonction de l'état interne présent et des entrées actuelles.

Nous pouvons classer (fig. 2.) plus précisement les machines séquentielles en fonction des capacités de leurs composants :

- Soit chaque cellule est une machine séquentielle définie par un quintuplet, toutes les cellules sont identiques. A chaque cellule est attachée une variable d'état. La fonction de transition globale est une composition des fonctions de transition locales. La boucle de rétro-action est implicite.
- Soit chaque cellule est définie par un triplet, la fonction de transfert est donc combinatoire. On distingue alors deux cas, selon la position des boucles de rétro-action :
    - Le traitement séquentiel est introduit par une boucle de rétro-action explicite sur chacune des cellules .
    - Des boucles de rétroaction externes sont introduites sur le schéma d'interconnexion et permettent de définir un sous ensemble de cellules pour le stockage de la mémoire. Dans ce dernier cas, la fonction de changement d'états est globale.

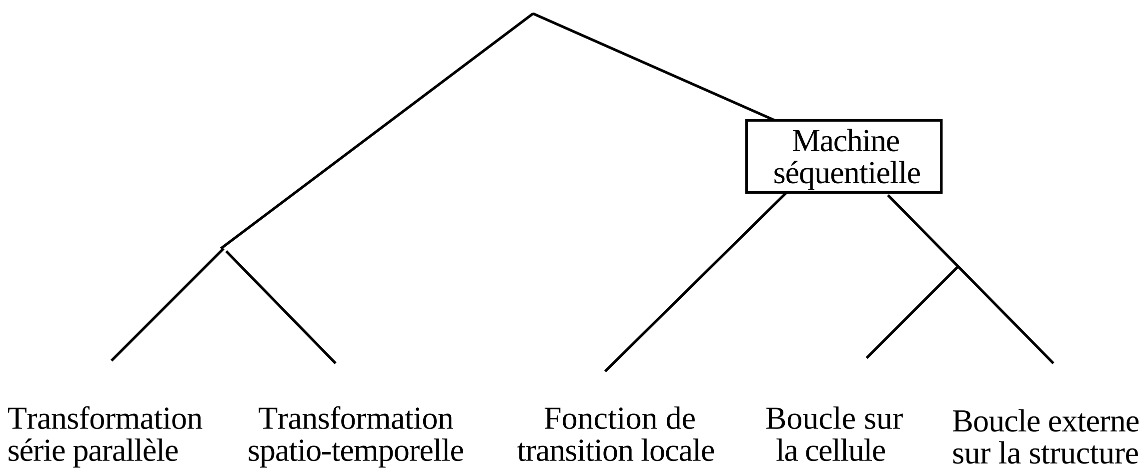


Figure 2. Classification des approches pour la résolution de problèmes séquentiels

Remarque : La classification que nous avons réalisé entre les différents modèles issus de la machine séquentielle ne se justifie que pour un certain niveau d'observation. A un autre niveau, il est possible de considérer tout modèle à boucles externes comme un modèle ne comprenant qu'une seule cellule avec boucle de rétro-action explicite. Réciproquement, tout modèle

composé de cellules avec boucle de rétro-action explicite peut être considéré comme une interconnexion de réseaux à boucle externe (chaque réseau ne comprenant qu'une seule cellule).

a/ Transformation série-parallèle

On réalise extérieurement au réseau une transformation série-parallèle. L'hypothèse est faite que l'on dispose, à un instant donné, d'une séquence de longueur fixée que l'on peut traiter en parallèle. Le comportement résultant est décrit par l'équation (o : la sortie, i : l'entrée) :

$$o(t) = F(i(t+n), i(t+n-1), ..., i(t), i(t-1), ..., i(t-n)), \quad \text{avec n petit en pratique.}$$

L'exemple d'application le plus connu est NETtalk. Un réseau de neurones multicouche, doté de la rétropropagation de gradient, apprend à prononcer un texte en langue anglaise. La prononciation d'une lettre dépend des 3 lettres précédentes et des 3 lettres suivantes. La taille de la séquence, appelée ici fenêtre, est donc de 7 lettres (fig.3.).

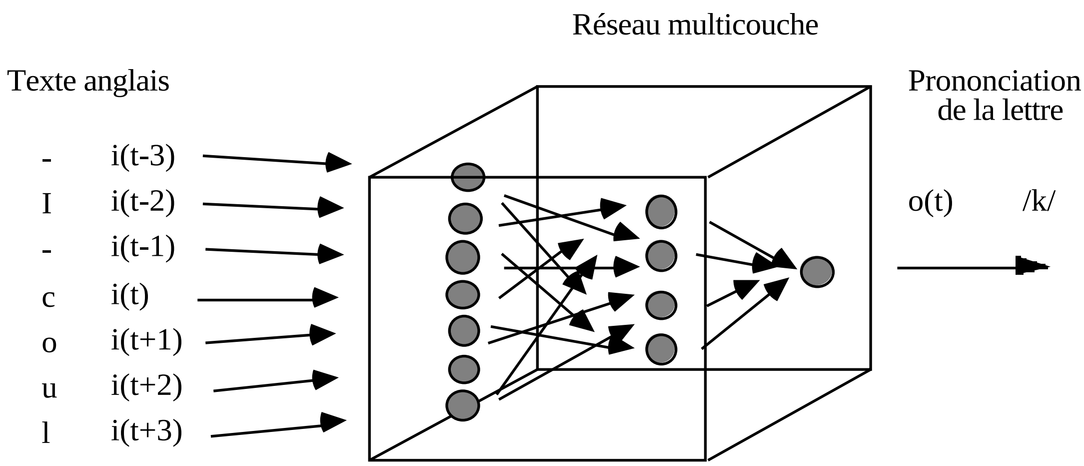


Figure 3. NETtalk : un exemple de transformation série-parallèle

Question : Quelles sont les limitations imposées par cette approche ?
Réponse : Un tel modèle ne permet pas de prendre en compte plus de 3 lettres précédentes. Comme le note Sejnowski, atteindre un haut niveau de performance (contrôle de l'intonation, ...) pour ce réseau nécessiterait la prise en compte d'une fenêtre plus large. Une entrée hors de la fenêtre ne pouvant pas intervenir sur la réponse du réseau, le passé lointain est inabordable par cette technique. La longueur maximale des séquences traitées doit être inférieure à la dimension de la fenêtre, qui reste faible en pratique.

b/ Approche spatio-temporelle

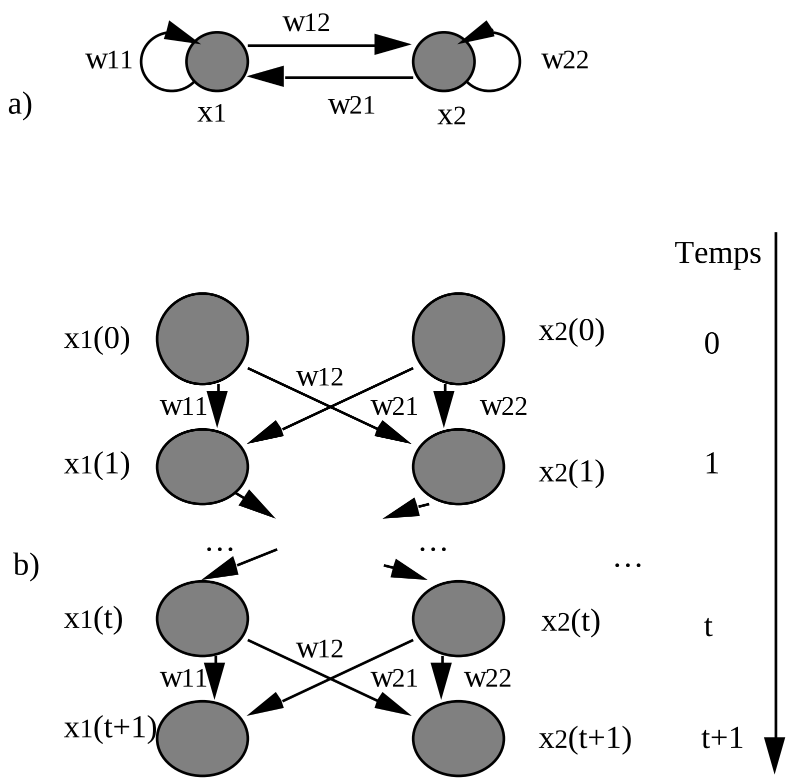


Figure 4. Approche spatio-temporelle

a) un réseau bouclé.

b) le réseau équivalent sans boucle sur une période de temps finie.

Les premieres études dans le cas des réseaux multicouches et de la rétropropagation de gradient sont dues à Rumelhart & al.. Ils proposent de construire pour chaque réseau à connexions récurrentes un réseau sans boucle montrant le même comportement sur une période de temps finie (fig. 4). Il s'agit en fait de dupliquer la structure dans l'espace.

c/ Fonction de transition locale

Les modèles biologiques de réseaux de neurones sont des exemples de modèles avec prise en compte du passé au niveau de la fonction réalisée par le neurone. Dans le modèle étudié par Zeigler, le neurone est binaire, son comportement est décrit par une équation de type : x(t+1) = f (t, a(t)). Il passe dans un état excité si la somme de ses entrées est supérieure à la valeur du seuil. Cette valeur de seuil est variable, elle dépend du temps écoulé depuis la dernière excitation (noté n). La valeur de seuil, dont le comportement est montré figure 4., est définie par une équation de la forme : seuil (t) = 1/ n .

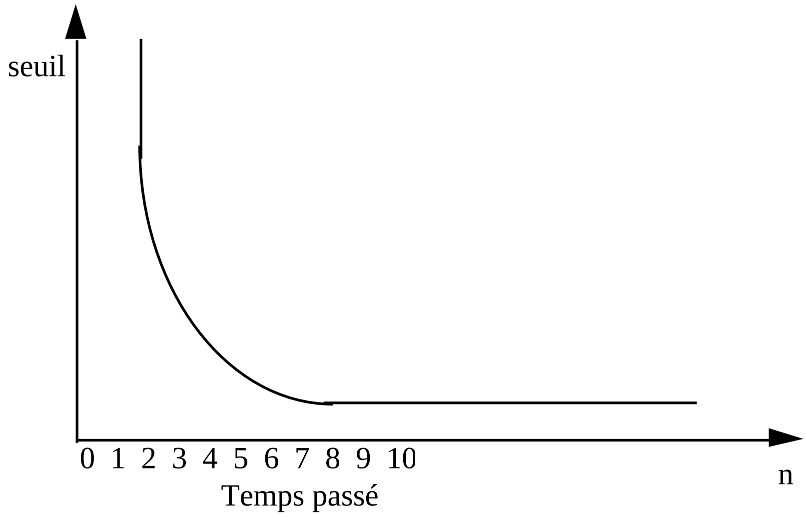


Figure 4. Une fonction de seuil typique

d/ Boucle sur le neurone

La fonction d'activation de chaque neurone est modifiée de façon à prendre en compte la présence de la boucle de rétroaction. Il s'agit en fait, dans le cas d'une boucle unique avec un retard égal à 1, de faire intervenir l'état présent x(t) pour le calcul de l'état futur.

Question : Donnez l'équation décrivant le comportement du réseau :

Réponse : $o(t+1) = F(i(t+1), x(t))$

Question : Proposez une équation de la fonction d'activation du neurone appartenant à cette approche ?

Réponse : $x(t+1) = (e^{(Ò.a(t) + k.x(t))} - 1) / (e^{(Ò.a(t) + k.x(t))} + 1)$

où Ò, k sont des constantes, a est la somme pondérée des entrées et f une fonction de type sigmoïde. k est le coefficient de prise en compte de l'état présent. Si k = 0 alors on retrouve la fonction d'activation classique. L'architecture du réseau est multicouche, l'algorithme d'apprentissage est dérivé de la rétroprogation de gradient. Bien que ce modèle, et d'autres du même type, montrent des capacités certaines dans l'apprentissage et la reconnaissance de séquences, la longueur des séquences traitées demeure faible.

e/ Boucles externes : Machine séquentielle connexionniste

Structure : Elle reprend la structure de la machine séquentielle en remplaçant les circuit digitaux par des réseaux multicouches (fig. 5.). Le premier bloc réalise la fonction de transition, le second réalise la fonction de sortie. Il y a une boucle de rétro-action sur le réseau réalisant la fonction de transition. Chaque neurone de la couche de sortie est rebouclé sur un neurone d’entrée. Tous les neurones sont identiques, les connexions se répartissent en deux classes selon la nature fixe ou plastique du poids de la connexion. Les connexions en provenance des entrées primaires et celles en provenance de la sortie du réseau réalisant la fonction de transition

sont de poids fixes (en traits gras). Les connexions entre les différentes couches dans chaque réseau sont plastiques, leur poids est modifié durant l'apprentissage. Tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante. Il n'y a pas de connexions entre neurones d'une même couche, ainsi que de connexion récurrente sur le neurone lui-même.

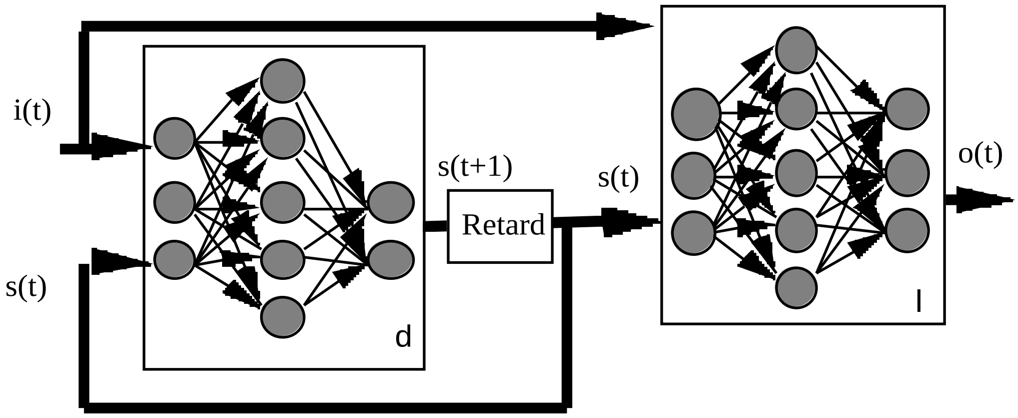


Figure 5. Machine séquentielle connexionniste

Fonctionnement : Le réseau réalisant la fonction de transition calcule à partir de l'état présent s(t) et de l'entrée actuelle i(t) l'état futur s(t+1). Le réseau réalisant la fonction de sortie calcule à partir de l'état présent et de l'entrée actuelle la sortie o(t).

Question : Donnez les équations définissant le comportement de chacun des réseaux :

Réponse : Réseau de transition $s(t+1) = \delta\,(s(t), i(t))$

Réseau de sortie $o(t) = \lambda\,(s(t), i(t))$

Le mode de comportement de cette machine est synchrone. Nous considérons que chaque boucle de rétroaction est dotée d'un poids égal à 1 et comporte un élément mémoire fictif, reproduisant sur sa sortie la valeur de son entrée, lorsqu'une impulsion d'horloge survient. Ce choix d'un comportement synchrone avec élément mémoire fictif n'est évidemment pas le seul envisageable. On peut, tout aussi bien, envisager un comportement asynchrone pur.

Question : Pourquoi avons nous choisi de remplacer les circuits combinatoires de la machine séquentielle par des réseaux multicouche ?

Réponse : Il nous intéressait de pouvoir modéliser des fonctions de transition et de sortie non linéaires. Ce n'est pas la seule possibilité, tout modèle neuronal est envisageable. Ainsi Kohonen a proposé une machine à états finis réalisée par une mémoire auto-associative. Les possibilités de ce modèle sont limitées par celles des mémoires associatives.

Le modèle de Jordan

L'architecture : Elle est multicouche, la dernière couche est rebouclée sur la première (fig. 6). Les cellules d'entrée se répartissent en deux groupes : les cellules de plan et les cellules d'état. Les cellules de sortie sont rebouclées sur les cellules d'état par des connexions de poids fixes, de même pour les cellules d'état qui rebouclent sur elle-même par des connexions de poids fixes. L'état interne est défini par l'ensemble des cellules de sortie et des cellules de plan.

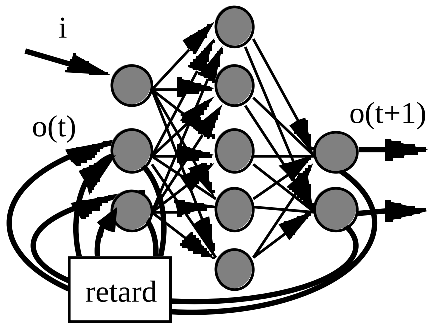


Figure 6. Réseau proposé par Jordan

Comportement : Une forme d'entrée i est appliquée sur les cellules de plan. Ce vecteur, appelé vecteur de plan, ne sera plus modifié pour la suite de la production de la séquence. Le vecteur d'état des cellules de sortie varie dans le temps du fait des connexions récurrentes sur les cellules d'état. Ces connexions modifient les valeurs d'activation des cellules d'état et imposent une entrée variable dans le temps au réseau multicouche. En utilisant des vecteurs de plan différents, le même réseau apprend plusieurs séquences différentes.

Question : Décrivez le comportement de ce modèle par une équation :

Réponse : $o(t+1) = F(i, o(t))$

L'algorithme d'apprentissage : C'est une généralisation de la rétropropagation de gradient. Plutôt que d'utiliser une valeur fixe pour les sorties désirées, des contraintes sont imposées sur ces valeurs. Une contrainte spécifie, par exemple, un intervalle pour la valeur de la sortie. Il y a apprentissage après chaque forme produite.

Résultats : Quelques simulations ont été réalisées. Par exemple, un réseau composé de 8 cellules d'état, 6 cellules de plan, 10 cellules cachées, 8 cellules de sortie, est capable d'apprendre à générer une séquence de 8 formes.

Le modèle d'Elman

L'architecture : Elle est légèrement différente de celle de Jordan. Il s’agit d’une structure multicouche (fig. 7.) où les boucles de rétro-action relient la couche cachée avec les cellules d’entrée. La couche d'entrée consiste en deux ensembles de cellules, les cellules de contexte et

les cellules d'entrée. Il y a autant de cellules cachées que de cellules de contexte. Chaque cellule cachée est reliée à une seule cellule de contexte par une connexion fixe, de poids égal à 1. L'état interne est défini par l'ensemble des cellules de la couche cachée.

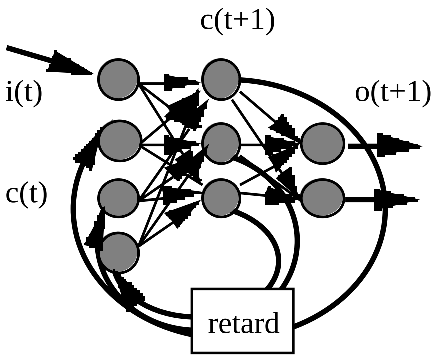


Figure 7. Réseau proposé par Elman

Comportement : La couche cachée a la tâche d'établir, à la fois, les associations de la fonction de sortie et les associations de la fonction de transition.

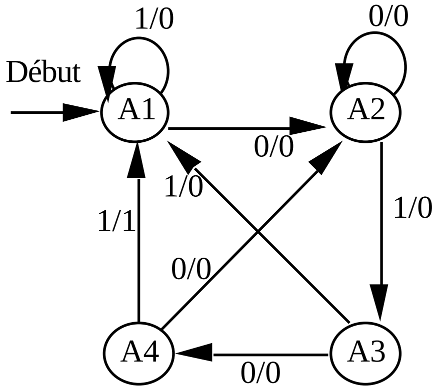
Question : Donnez les équations représentatives du comportement de ce modèle sachant qu'il nécessite d'introduire la fonction G réalisée par la première couche de poids du réseau.

Réponse : o(t+1) = F(i, c(t)) , c(t+1) = G(i(t), c(t))

A chaque instant, une sortie est calculée sur la base de l'état interne (contexte) et de l'entrée présente. Il y a copie du vecteur d'activation de la couche cachée sur la couche de contexte. De cette façon, le résultat intermédiaire du traitement à la date précédente (t-1) peut infuencer le traitement à la date actuelle (t).

L'algorithme d'apprentissage : C'est la rétropropagation de gradient. Les vecteurs d'activation de la couche cachée développés par le réseau sont significatifs du problème soumis. Ces vecteurs d'activation sont les représentations internes construites par le réseau.

Résultats : Dans le cas où les séquences d'apprentissage ont été générées en accord avec le graphe d'états finis d'un automate, l'apprentissage permet de réaliser la synthèse de cet automate. Le codage des états internes du réseau est représentatif des noeuds du graphe d'états (fig. 8.).



Entrée : 11000111010001011010001110101
Sortie : 00000000000000010000000000001

Figure 8. Le graphe d'états d'un automate détecteur de la séquence 0101 (longueur 4). Sur chaque transition, on a indiqué la valeur de l'entrée et la sortie correspondante. Une séquence d'apprentissage (Entrée, Sortie) est présentée au dessous. Un millier d'itérations d'apprentissage est suffisant pour que le réseau syjnthétise l'automate correspondant aux séquences d'apprentissage.

Le réseau est alors capable de classer sans erreur des séquences de longueur quelconque comme appartenant ou non à l'automate. Le réseau de neurones se compose d'une couche de 4 cellules d'entrée, 3 cellules de contexte, 3 cellules cachées et 2 cellules de sortie, alors que le graphe d'état montre 4 états, 8 transitions et 2 valeurs d'entrées (notées sur les arcs). Ces précisions techniques permettent de situer les applications envisageables, les performances étant fortement liées à la taille de la grammaire.

Question : En utilisant le formalisme généré par la machine séquentielle connexionniste, que peut-on déduire du comportement, des potentialités des modèles de Jordan et d'Elman. Décrivez, en les dessinant, ces deux modèles comme des machines séquentielles connexionnistes particulières. A partir des structures de réseaux définies pour réaliser les fonctions de transition et de sortie, il est possible de connaître les classes de fonctions réalisables. On appréhende ainsi les applications que ces différentes machines séquentielles sont capables de traiter.
Réponse : La machine séquentielle connexionniste permet de représenter les modèles de Jordan et d'Elman. Le grand intérêt de telles représentations est la possibilité de prédire et d'expliquer le comportement de chacun de ces modèles, les fonctions de transition et de sortie y étant plus explicites.

Le modèle de Jordan (fig. 9.) est une machine d'états connexionniste. La fonction de transition est réalisée par un réseau multicouche (3 couches). Il y a une connexion récurrente de

chacun des neurones d'états internes sur lui-même. Ce modèle se comporte comme une machine d'état asynchrone. Pour un vecteur d'entrée donné, on observe l'évolution de la machine, qui passe par plusieurs états transitoires pour éventuellement aboutir dans un état stable.

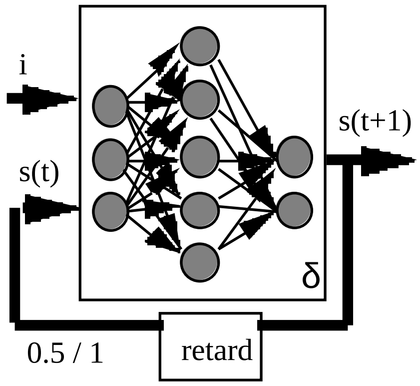


Figure 9. Le modèle de Jordan : une machine d'état connexionniste

Le modèle d'Elman (fig. 10.) correspond à une machine séquentielle connexionniste où les réseaux réalisant les fonctions de transition et de sortie sont des réseaux de neurones à deux couches. Il n'y a pas d'entrée primaire sur la fonction de sortie, c'est donc un modèle de type machine de Moore. La fonction de transition et la fonction de sortie sont apprises. Cependant, comme il n'y a qu'une seule couche de poids modifiables pour chaque fonction, les possibilités au niveau des fonctions réalisables sont réduites. Seules peuvent être réalisées des fonctions linéairement séparables. Par rapport à la machine séquentielle connexionniste, ce modèle ne peut pas réaliser d'associations non-linéaires au niveau de la fonction de transition.

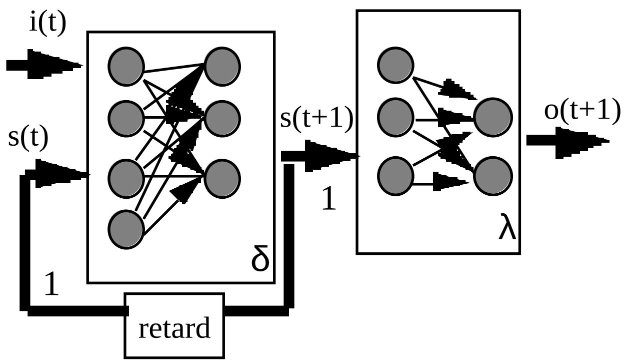


Figure 10. Le modèle d'Elman : une machine séquentielle connexionniste dégénérée

Le modèle de Jordan et celui d'Elman présentent certaines limitations que nous avons explicitées. L'approche suivie par ces auteurs est empirique : ayant imaginé une structure de réseau, ils vérifient qu'elle répond, ou non, au problème. La démarche qui a procédé au

développement de la machine séquentielle connexionniste est inverse. Une structure générale pour la résolution de problèmes de nature séquentielle est proposée, structure que l'on adapte pour des applications particulières. De fait, la machine séquentielle connexionniste est une généralisation des réseaux de neurones séquentiels.

3 Construction d'une taxonomie des modèles de réseaux neuronaux

Huit critères sont utilisés pour répertorier les modèles :

temps discret / continu, variables discrètes / continues, déterministe / stochastique, modèle autonome / non autonome, invariance / non invariance par rapport au temps, combinatoire / séquentiel, modèle instantané / non instantané, synchrone / asynchrone.

Question : Dans une simulation à temps discret, les évènements se passent à des instants précis, séparés dans le temps. Comment caracterisez-vous les modèles que vous connaissez ?
Réponse : Aujourd'hui, la plupart des réseaux de neurones sont des modèles à temps discret. L'unique raison semble être la facilité de programmation et de simulation.

Question : Les variables descriptives du modèle peuvent être soit discrètes, soit continues. Comment caracterisez-vous les modèles que vous connaissez ?
Réponse : Il ne s'agit pas là d'un critère trés utile. Les réseaux neuronaux contiennent souvent des variables continues : les poids des connexions. Notons cependant qu'un réseau dont les variables de sortie sont continues peut fournir une infinité de réponses. Inversement un réseau dont les sorties sont discrètes ne peut fournir qu'un nombre fini de réponses. Cette remarque est particulièrement importante dans le cas des variables d'état des réseaux séquentiels (cf machine séqentielle connexionniste).

Question : Un modèle est stochastique si sa description contient des variables aléatoires, déterministe dans le cas contraire. Comment caracterisez-vous les modèles que vous connaissez ?
Réponse : ARP est un modèle stochastique, que l'apprentissage rend déterministe.

Question : Un modèle autonome est un système sans interaction avec le monde réel. Comment caracterisez-vous un réseau de neurones autonome que vous connaissez ?
Réponse : Un modèle autonome, dans le cas des réseaux de neurones, est un réseau sans entrée. Les variations de la sortie résultent de phénomènes internes générés par des neurones oscillatoires par exemple.

Question : Un modèle est invariant par rapport au temps si le temps n'intervient pas comme argument des règles d'interactions. Celles-ci sont de fait les mêmes à n'importe quelle date. Comment caracterisez-vous les modèles que vous connaissez ?

Réponse : La plupart des réseaux sont des modèles invariant par rapport au temps. Dans le domaine neuromimétique, les choses sont souvent différentes. Par exemple, le neurone proposé par Zeigler (exercice précédent) est doté d'un comportement binaire dont le seuil varie au cours du temps. De cette façon, le neurone est insensible à l'arrivée de nouveaux stimuli après excitation (simulation de la période réfractaire).

Question : La sortie d'un modèle instantané est obtenue dès la présentation d'une nouvelle entrée. Un modèle séquentiel est nécessairement non instantané. Imaginez des modèles de réseaux combinatoires non instantanés
Réponse : Il s'agit des réseaux combinatoires avec retards sur les connexions.

Question : Dans un modèle synchrone, toutes les variables d'état évoluent à la même date. Ce critère s'applique aux modèles séquentiels ou combinatoires non instantané. Comment caracterisez-vous les modèles que vous connaissez ?
Réponse : La plupart des réseaux sont considérés (implicitement) comme synchrones. Remarquons que l'on peut construire une autre catégorie selon la nature implicite ou explicite du temps dans les modèles. Dans la plupart des modèles, le temps est implicitement confondu avec la notion d'itération. Cette classe de modèles à temps implicite regroupe ensemble les modèles combinatoires instantané et séquentiels synchrones à temps discret. Cependant, cette catégorisation est moins précise.

4 Coopération multi-réseaux

La lecture des ouvrages recensants les applications actuellement développées permet de constater que la très grande majorité des applications impliquent au maximum 300 neurones. Ceux-ci sont habituellement répartis comme suit : 200 neurones d'entrées, 70 neurones cachés et 30 neurones de sortie, au sein d'un unique réseau muni d'un algorithme d'apprentissage de type rétropropagation de gradient. Le traitement réalisé par le réseau est l'apprentissage d'une fonction de mise en correspondance entre l'espace d'entrée et l'espace de sortie. La généralisation est le résultat d'une interpolation non-linéaire effectuée sur les exemples d'apprentissage. Ainsi, nous sommes aujourd'hui capables d'analyser le traitement réalisé par ce type de réseau (analyse en composantes principales sur la première couche, etc). L'application d'un modèle de réseau unique, avec un petit nombre de neurones, ne permet de résoudre que des probèmes simples.

Question : La résolution de problèmes complexes impose d'augmenter le nombre de neurones. Mais quelles sont les causes qui restreignent le nombre de neurones mis en oeuvre ? Comment les contourner ?
Réponse : Deux parmi les causes recencées sont relatives aux temps de calculs (pour l'apprentissage) et à l'abscence d'études sur le comportement des algorithmes d'apprentissage

sur "grands" réseaux (de l'ordre de 100 000 neurones). Ces deux limitations peuvent être contourné en changeant de niveau d'organisation. Manipuler des interconnexions de réseaux augmente le nombre de neurones mis en jeu. N'oublions pas l'essentiel cependant, l'élément de base du système reste le neurone (et non le réseau). Il faut concevoir ce changement de niveau comme une facilité pour la construction et l'apprentissage de grands réseaux neuronaux.

Question : Le développement des études sur la coopération multi-réseaux est une approche possible pour résoudre des problèmes complexes. La résolution d'un problème par un réseau de neurones consiste à trouver par apprentissage la bonne relation d'entrée/sortie. Lorsque cette relation est complexe, le problème doit être décomposé en sous-problèmes et ainsi de suite jusqu'à aboutir à un ensemble de sous-problèmes terminaux pour lesquels on sait trouver une solution. L'ensemble des comportements et leur schéma d'interconnexions constitue une décomposition structurelle hiérarchisée du problème. Proposez deux réalisations pratiques de cette approche.

Réponse : Dans le premier cas, chacun des comportements des sous-problèmes terminaux est réalisé par un réseau de neurones, que nous appelons réseau de base. On détermine donc pour un problème complexe donné, d'une part la structure du système et d'autre part, les comportements que doivent réaliser chacun des réseaux de bases. L'apprentissage peut être réalisé en contexte, à partir de la seule relation globale d'entrée/sortie, ou hors contexte si l'on connait pour chacun des réseaux de base le comportement à réaliser. En d'autres termes, on peut donc considérer que le problème de la construction de réseaux de neurones hiérarchisés se décompose en : définition de structures adaptées à des classes de problèmes spécifiques, choix du type et de la structure des réseaux de neurones apparaissant dans la hiérarchie, mise en oeuvre de la technique d'apprentissage hors contexte et/ou en contexte. Les travaux réalisés sur les machines séquentielles connexionnistes constituent une première approche à l'étude de la combinaison de plusieurs types de réseaux neuronaux. Dans ce cas, il s'agit de coupler deux réseaux de neurones correspondant respectivement à la fonction de transition et à la fonction de sortie. Chacune de ces fonctions est réalisée par un réseau de neurones multicouche. La structure est fixée a priori. Ce modèle réalise une synthèse automatique d'un automate à partir d'exemples de séquences. Dans le cas où le graphe d'état de l'automate à synthétiser par le réseau n'est pas connu, on ne peut pas réaliser un apprentissage hors contexte de chacun des réseaux.

Dans le second cas, il s'agit d'utiliser le travail réalisé par l'analyste/programmeur pour la résolution du problème en remplaçant chaque bloc d'instructions par un réseau de neurones. Le réseau se comporte alors comme la fonction prescrite par le code (phase d'apprentissage). De plus, la réalisation neuronale permet d'envisager une gestion efficace de situations imprévues

(conditions aux limites, dérive des données d'entrée, etc). L'un des modèles de réseau de neurones qui semble le plus adapté à cette situation est la machine séquentielle connexionniste. Chaque procédure est une machine séquentielle particulière que l'on remplace par la machine séquentielle connexionniste correspondante. Celle-ci est à même de synthétiser tout comportement séquentiel, qui apparait notamment dans la gestion des boucles.

Remarque : Devant de telles architectures neuronales, nous ne sommes plus capables d'effectivement comprendre le comportement du réseau. Sommes nous de ce fait en train de générer un comportement intelligent, de par sa nature incompréhensible ?

15 Annexes

1 Carte auto-organisatrice

Certains neurones peuvent répondre de moins en moins activement à un stimulus lorsque celui-ci se présente fréquemment. Une fréquence de sélection est introduite, qui s'incrémente ou se décrémente selon que le neurone est le foyer d'activation (cluster) ou non. Ensuite, la corrélation des neurones aux entrées est pondérée par un facteur dépendant de la fréquence d'activation.

Algorithme d'apprentissage modifié pour les cartes auto-organisatrices :

1/ Initialisation des poids à des valeurs aléatoires autour d'une valeur centrale.
Initialisation des fréquences : pour tout neurone i, freq[i] = constante.
Initialisation des gains $\mu(0)$ et $ß(0)$.
Choix de nombre_itérations.

2/ Présentation d'une entrée $E_l = (e_1, e_2)$.

3/ Calcul de la distance de chacun des neurones par rapport à e_1 et e_2

$x_j = |w_{j1} - e_1| + |w_{j2} - e_2|$ (∗ calcul de la distance ∗)

$x_j = x_j.(freq[j] + a / nombre_neurones)$ (∗ pondération de la distance par la fréquence ∗)
a : constante réelle positive.

4/ Sélection du neurone le plus proche : $Min(x)= x_i$

5/ Modification des poids pour le neurone choisi (i) et ses 4 plus proches voisins (k).

$w_{i1} = w_{i1} + \mu(t). (e_1 - w_{i1})$

$w_{i2} = w_{i2} + \mu(t).\ (e_2 - w_{i2})$

$w_{k1} = w_{k1} + ß(t).\ (e_1 - w_{k1})$

$w_{k2} = w_{k2} + ß(t).\ (e_2 - w_{k2})$

$\mu(t) > 0$ et $0 < ß(t) \leq a(t)$

6/ Mise à jour des fréquences (b > 0) :

Si i est le neurone choisi : freq[i] = freq[i] + b · ((1 - freq[i]) / nombre_neurones)

sinon : freq[j] = freq[j] + b · ((0 - freq[j]) / nombre_neurones)

7/ Décrémentation des gains µ et ß :

$\mu(t+1) = \mu(t) - \mu(0)$ / nombre_itérations

$ß(t+1) = ß(t) - ß(0)$ / nombre_itérations

8/ Tant que $\mu(t) > 0$: Retour à l'étape 2 et sélection de l'exemple suivant dans la base d'apprentissage.

2 Rétropropagation de gradient

Il faut trouver une configuration de poids qui minimise un critère d'erreur. On définit donc une fonction de coût :

$$C(W) = M[C_l(W)] = M[\textstyle\sum_j e_{lj}^2(W)] \text{ avec } e_{lj} = (d_{lj} - x_{lj})$$

où, j indique un numéro d'indice pour les neurones de sortie et l indique un exemple d'apprentissage. M est l'opérateur de moyennage, c'est une estimation de la moyenne temporelle dans le cas stochastique. On réalise donc la moyenne des erreurs obtenues pour chacun des exemples de la base d'apprentissage.

L'algorithme du gradient permet de trouver une configuration minimale pour la fonction de coût, il suffit d'appliquer de manière itérative la formule :

$$W(t+1) = W(t) - \mu \,.\, \partial C(Wt)$$

où $\partial C(t)$ ets la dérivée partielle de C par rapport à tous les w_{ij}.

Cet algorithme nécessite une fonction continue, non-linéaire et différentiable comme fonction de transfert du neurone.

$$a_i = \textstyle\sum_j w_{ij} \,.\, x_i \quad ; \qquad f(a_i) = (e^{Òa_i} - 1) / (e^{Òa_i} + 1)$$

La linéarité de l'opérateur de moyennage permet d'effectuer tous les calculs sur les valeurs instantanées de la fonction de coût.

$$\partial C/\partial w_{ij} = \partial C / \partial a_i \,.\, \partial a_i / \partial w_{ij} \quad \text{(dérivation des fonctions composées)}$$

Sachant que $a_i = \sum_j w_{ij} . x_i$ alors $\partial C/\partial w_{ij} = \partial C / \partial a_i . x_j$

Il nous reste donc à calculer les $\partial C / \partial a_i$ que nous notons $-y_i$

Si i est une cellule de sortie : $y_i = -\partial C / \partial a_i = -\partial \sum_q e^2_q$

Après dérivation, $y_i = -\sum_q \partial e^2_q / \partial x_i . \partial x_i / \partial a_i$ ($\partial e^2_q / \partial x_i$ est nulle si $q \neq i$ pour toute cellule de sortie, indépendantes de fait).

d'où $y_i = -\partial e^2_i / \partial x_i . \partial x_i / \partial a_i = -2e_i . \partial e_i / \partial x_i . \partial x_i / \partial a_i$

Sachant que pour les cellules de sortie $e_i = (d_i - x_i)$, alors $\partial e_i / \partial x_i = -1$

Résultat : $y_i = 2e_i . f'(a_i)$

Si i est une cellule cachée (ou d'entrée) : $y_i = -\partial C / \partial a_i$

Posant l'hypothèse que tous les y_k auxquelles la cellule i envoie sa sortie sont connus alors

$$y_i = \sum_k (-\partial C / \partial a_k . \partial a_k / \partial a_i) = \sum_k (y_k . \partial a_k / \partial x_i . \partial x_i / \partial a_i)$$

Soit $y_i = \sum_k (y_k . w_{ki}. f'(a_i)) = f '(a_i). \sum_k(w_{ki}.y_k)$

Le calcul du gradient attaché à une cellule utilise les poids qui les relient aux cellules avals ainsi que des gradients correspondants y_k (fig. 1).

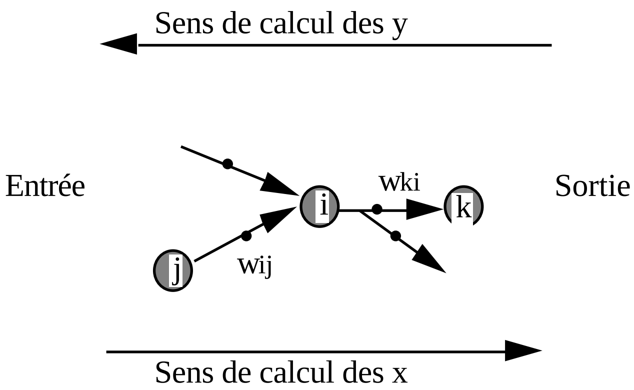


Figure 1. Définition d'un sens de propagation de l'information opposé pour le calcul des x et des y.

3 Algorithme d'apprentissage par pénalité/récompense (A_{RP})

Il a été originellement proposé par Barto et Anandan. Il s'applique à toute architecture de réseau, la seule contrainte étant d'avoir des neurones de sortie stochastiques, gouvernés par la règle habituelle :

$$\text{Probabilité } (x_i = \pm 1) = g(a_i) = (1 + \exp(\pm 2 Ò a_i))^{-1} \text{ où } a_i = \sum_j w_{ij} . x_i$$

La valeur x_i (binaire) des neurones de sortie est tirée au sort avec une probabilité que le neurone se trouve dans l'état +1 fonction de la somme pondérée des entrées a_i. Cette fonction g est une

sigmoïde dont les valeurs asymptotiques sont 0 et +1. Ò est un paramètre qui détermine la valeur de la pente de la courbe.

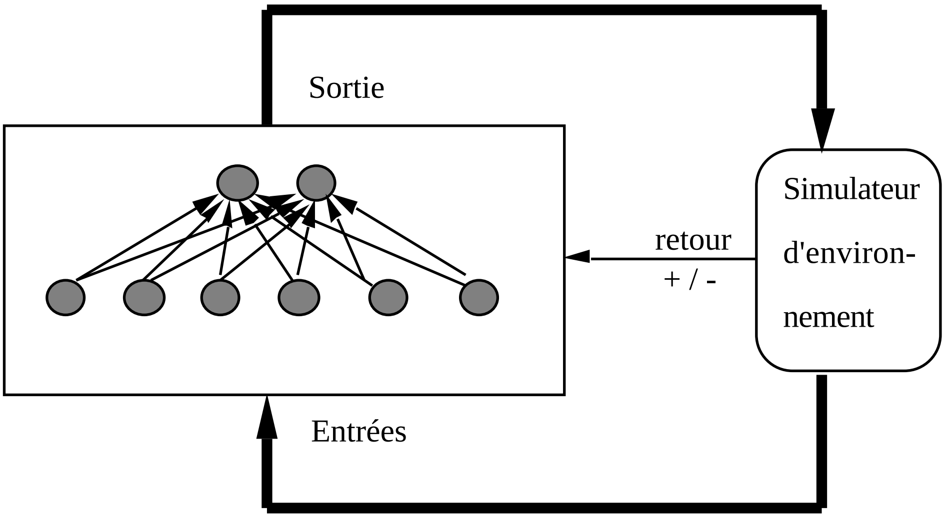


Figure 2. Algorithme d'apprentissage par pénalité/récompense.

Algorithme

1/ Dans une première étape, les poids sont initialisés à de petites valeurs aléatoires qui placent les probabilités des neurones de sortie autour de 0.5.

2/ une entrée E = (e_1, … e_n) est présentée,

3/ une sortie correspondante possible x_i est calculée pour chaque neurone,

4/ La sortie globale produite est analysée de façon à générer un signal de retour r, positif ou négatif, et une sortie cible (désirée) est choisie :

$d_i = x_i$ si r = +1 (récompense)

$d_i = -x_i$ si r = -1 (pénalité)

5/ les poids sont modifiés par une méthode de gradient en prenant pour sortie effective la moyenne $\langle x_i \rangle$ des sorties que l'on aurait obtenues si l'on avait effectué un très grand nombre de présentation de la même entrée. Rappelons que le comportement stochastique des neurones de sortie autorise l'obtention de sorties différentes pour une même entrée et avec la même configuration de poids. La moyenne est donnée par la formule :

$\langle x_i \rangle = (+1) . g(a_i) + (-1) . (1 - g(a_i))$ = tangente_hyperbolique (Òa_i)

L'erreur considérée est alors :

$erreur_i = d_i - \langle x_i \rangle$

La modification des poids est réalisée par la classique méthode du gradient :

$\partial w_{ij} = \mu . r . erreur_i . x_j$

En général, μ dépend de r et est pris 10 à 100 fois plus grand (μ^+) pour r = +1 que pour r = -1 (μ^-). La règle globale est donc finalement :

$\partial w_{ij} = \mu^{+} . (x_i - \langle x_i \rangle) . x_j$ si r = +1 (récompense)

$\partial w_{ij} = \mu^{-} . (-x_i - \langle x_i \rangle) . x_j$ si r = -1 (pénalité)

6/ Puis, tant que la sortie du réseau n'a pas produit une séquence satisfaisante suffisamment longue, retour à l'étape 2.

4 Approximation de fonction par réseau de neurones

Théorème :

"Pour toute valeur e > 0 et toute fonction f de L_2 telle que f : $[0, 1]^n \to R^m$, il existe un réseau multicouche à trois couches doté de l'algorithme d'apprentissage de la rétropropagation de gradient qui peut approximer f avec une mesure de l'erreur au carré inférieure à e."

L2 inclut toutes les fonctions qui peuvent apparaitre dans une application pratique (continues et discontinues). Il faut remarquer que ce théorème ne donne aucune indication sur le nombre de neurones dans la couche cachée et d'autre part, il présente les réseau à une couche cachée comme un minimum. En pratique, des réseaux avec deux ou trois couches cachées peuvent se révéler plus performants. D'autre part, ce théorème garantit qu'un réseau à trois couches avec les poids adéquats peut approximer n'importe quelle fonction de L2, mais il n'exprime aucune idée sur la façon dont ces poids pourraient être calculé.

5 La simulation dirigée par les évènements

La simulation évènementielle est une méthode peu utilisée dans le domaine des réseaux de neurones artificiels, bien qu'elle soit très employée en CAO (Conception Assistée par Ordinateur) des circuits. La simulation évènementielle présente de multiples avantages par rapport au calcul matriciel (matrice des poids) classiquement utilisée, dont :

- Gain en facilité de conceptualisation : les neurones et synapses sont effectivement représentés comme des entités.
- Gain de temps de calcul : seuls les neurones ayant changé d'états créés un évènement. De fait, le minimum de calcul est effectué. Rappelons que le calcul matriciel oblige de recalculer à chaque interaction la valeur des états de tous les neurones du réseau. Ceci est particulièrement pénalisant dans le cadre des réseaux multicouches sans boucle où les couches sont activées les unes après les autres (et une seule à la fois).
- Gain de place mémoire : la structure de données décrivant le modèle est minimale, en effet seules les neurones et les connexions existants sont décrits. La technique classique (matricielle) impose de manipuler des matrices de connexions très creuses. Ainsi dans le cas d'un réseau de neurones multicouche à connexions complète intercouches sans boucles comprenant n cellules, la taille de la matrice est de n^2 ; alors qu'il n'existe qu'environ 0.1 n^2 connexions effectives. Pour des réseaux de grandes dimension, le gain de taille mémoire est considérable.

Les éléments de base d'un simulateur dirigé par les événements sont l'échéancier et les évènements. L'échéancier est une structure dans laquelle sont stockés les événements. A chaque événement est associé une date. L'échéancier assure le classement des événements en fonction de leur date afin que soit traité en premier l'événement le plus proche dans le temps. Les évènements, dans notre cas, sont au nombre de deux. Un premier type d'évènement (ev1) correspond à la transmission d'information depuis un neurone, ou une entrée, ayant changé d'état. Il peut être vu comme le cheminement d'un potentiel d'action le long de l'axone. Cet événement permet l'introduction explicite au niveau de la synapse du concept de retard (encore peu exploité au sein des modèles neuromimétiques développés actuellement). Le second type d'évènement (ev2) calcule l'état du neurone en réalisant, à chaque instant, la somme des informations arrivées. Cet évènement génère, éventuellement, de nouveaux évènements (ev1).

16 Bibliographie

Chapitre 1

J. Anderson, E. Rosenfeld, *Neurocomputing : Foundations of research*, MIT Press, Cambridge, Second printing, ISBN 0-262-01097-6, 1988.

On trouve dans cet ouvrage tous les articles importants tels que :

W. James, *Psychology (Briefer Course)*, New york: Holt, Chapter XVI, "Association,"pp. 253-279, 1890.

W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics 5: 115-133, 1943.

D. Hebb, *The Organization of Behavior*, New York: Wiley, 1949.

F. Rosenblatt, "The Perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review 65 : 386-408, 1958.

B. Widrow and M. Hoff, "Adaptive switching circuits," 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104, 1960.

J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities," Proceedings of the National Academy of Sciences 79 : 2554-2558, 1982.

D. Ackley, G. Hinton and T. Sejnowski, "A learning algorithm for Boltzmann machines," Cognitive Science 9 : 147-169, 1985.

D. Rumelhart, G. Hinton & R. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing, Vol. 1.*, D. Rumelhart and J. McClelland Eds. Cambridge: MIT Press, pp. 318-362, 1986.

DARPA, *DARPA Neural Network Study*, Chap. 8. AFCEA International Press, 1988.

M. Minsky and S. Papert, *Perceptrons*, Expanded Edition. MIT Press, 1988.

Chapitre 2

Le Cerveau, Bibliothèque de la revue Pour la Science, ISBN 2-902918-24-7, 1982.

R. Masland, "L'architecture fonctionnelle de la rétine," revue Pour la Science, pp.94-104, février 1987.

J. P. Changeux, *L'homme neuronal*, Collection Pluriel, Fayard, ISBN 2-01-009635-5, 1983.

Chapitre 3

B. Zeigler,*Theory of Modelling and Simulation*, Malabar: Robert E. Krieger Publishing Company Inc, 1976.

Chapitre 4

D. Hebb, cf. chp. 1.

F. Rosenblatt, cf. chp. 1.

D. Rumelhart, G. Hinton & R. Williams, *Parallel Distributed Processing*, Vol. 1. Cambridge: MIT Press, 1986.

J. L. Mc Clelland and D. E. Rumelhart, *Explorations in Parallel Distributed Processing, a Handbook of Models, programs, and Exemples*, MIT Press, Cambridge, 1988.

Chapitre 5

C. Jutten, *Calcul neuromimétique et traitement du signal, analyse en composantes indépendantes*, thèse de Doctorat d'état, INPG, Grenoble, 1987.

T. Kohonen, *Self-Organization and Associative Memory* ; Second Edition, Springer Series in Information Sciences, Vol. 8, Springer Verlag, Berlin , ISBN 3-540-18314-0, 1987.

Chapitre 6

T. Kohonen, cf. chp. 5.

Y. Coiton, Thèse de Doctorat, Uiversité d'Aix-Marseille II, juillet 1992.

Y. Coiton, J. C. Gilhodes, J. L. Velay and J. P. Roll, "A neural network model for the intersensory coordination involved in goal-directed movements," Biological Cybernetics, n° 66, pp. 167-176, 1991.

S. Dellaporta, *Compression d'images par carte auto-organisatrice*, rapport de DEA, USTL, Montpellier, juillet 1991.

Chapitre 7

G. A.Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," Proceedings IEEE, March 1988.

S. Sehad, *Réseaux neuronaux "ART", application à la coalescence de données en génétique*, rapport de DEA, Universitté de Montpellier II, 1992.

Chapitre 8

C. Touzet et O. Sarzeaud, "Application d'un algorithme d'apprentissage par pénalité/récompense à la génération de formes locomotrices héxapodes", Journées de Rochebrune, AFCET I.A., 20-24 janvier 1992.

A. G. Barto & P. Anandan, "Pattern Recognizing Stochastic Learning Automata," IEEE Transactions on Systems, Man and Cybernetics, SMC-15 : 360-375, 1985.

R. A. Brooks, "A robot that walks: Emergent behaviors from a carefully evolved network," Neural Computation 1(2) : 253-262, 1989.

Chapitre 9

Y. le Cun, *Modèles Connexionnistes de l'Apprentissage*, Thèse de Doctorat, Université Pierre et Marie Curie, Paris, France, 1987.

D. Rumelhart, G. Hinton & R. Williams, cf. chp 1.

N. Giambiasi, R. Lbath & C. Touzet, "Une approche connexionniste pour calculer l'implication floue dans les systèmes à base de règles", Proc. Neuro-Nîmes 89, Nîmes, France, EC2 (eds), ISBN 2-906899-29-1, novembre 1989.

P. Handcock, "Data representation in neural nets: an empirical study," Proc. of the 1988 Connectionist Models Summer School, D. Touretzky & al. Eds, Morgan Kaufmann Publishers, ISBN 1-55860-033-07, 1988.

Chapitre 10

A. Brons, G. Rabatel, F. Sévila et C. Touzet, "Evaluation de la qualité des plantes en pots par un réseau multicouche, assisté par des méthodes statistiques," Proc. Neuro-Nîmes 91, Nîmes, France, EC2 (eds), ISBN 2-906899-67-4, novembre 1991.

F. Blayo et P. Demartines, "Algorithme de Kohonen, Application à l'analyse de données économiques," Bulletin SEV/VSE, Zurich, Suisse, 13 mars 1992.

O. Sarzeaud, Y. Stephan et C. Touzet, "Application des cartes auto-organisatrices à la génération de maillage aux éléments finis", Proc. Neuro-Nîmes 90, Nîmes, France, EC2 (eds), ISBN 2-906899-47-X, novembre 1990.

Darpa, cf. chp. 1.

Chapitre 11

C. Touzet et N. Giambiasi, *S.A.C.R.E.N. : Système d'Aide au Choix d'un Réseau de Neurones*, rapport final du contrat ANVAR n° A8801006JAL, Juillet 1989.

D. Rumelhart, G. Hinton & R. Williams, cf. chp. 4.

Chapitre 12

F. Blayo, *Une implantation systolique des algorithmes connexionnistes*, Thèse de Doctorat n° 904, EPFL, Lausanne, Suisse, 1990.

Chapitre 13

R. D. Beer, *Intelligence as Adaptive Behavior, An Experiment in Computational Neuroethology*, Perspectives in Artificial Intelligence, Vol. 6, Academic Press, Inc., ISBN 0-12-084730-2, 1990.

P. Maes, *Designing Autonomous Agents*, MIT/Elsevier, ISBN 0-262-63135-0, 1990.

Chapitre 14

C. Touzet,*Contribution à l'étude et au développement de modèles connexionnistes séquentiels,* Thèse de Doctorat, Université des Sciences et Techniques du Languedoc, Montpellier, France, 1990.

Chapitre 15

Z. Kohavi,*Switching and Finite Automata Theory* , Second Edition, Computer Science Series, McGraw-Hill, 1978.

R. Hecht-Nielsen, *Neurocomputing*, Redwood City: Addison-Wesley Pub. Co, ISBN 0-201-09355-3, 1989.

17 Informations pratiques

Ouvrage généraux

Le groupe de recherche PDP (Parallel Distributed Processing), à l'initiative de J. McClelland et D. Rumelhart, a publié deux ouvrages de référence (aussi appelé le PDP ou "la bible") sur les modèles connexionnistes :

D. E. Rumelhart, J. L. Mc Clelland and the PDP Research Group, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Vol 1 : Foundations, Vol 2 : Psychological and Biological Models*, MIT Press, Cambridge, Massachussets, London, England 1986.

Quelques années plus tard, ces deux auteurs ont rédigés un troisième ouvrage consacré à l'étude des systèmes parallèles distribués qui propose maints programmes et exemples. Il est destiné à servir de support pour l'acquisition d'une certaine expérience pratique aux amateurs de réseaux neuronaux. En même temps que le livre est fourni un ensemble de programmes pour la simulation des différents modèles. Ces logiciels sont écrits en C (le code source est fourni) sur deux disquettes de $5^{1}/4$" au format MS-DOS et sont donc utilisables sur matériel PC. L'utilisateur est à même de modifier les programmes afin de les adapter à ses désirs.

J. L. Mc Clelland and D. E. Rumelhart, *Explorations in Parallel Distributed Processing, a Handbook of Models, programs, and Exemples*, MIT Press, Cambridge, 1988.

J. Anderson, E. Rosenfeld on édité un ouvrage qui regroupe pratiquement tous les articles de recherche important jusqu'en 1987 :

J. Anderson, E. Rosenfeld, *Neurocomputing : Foundations of research*, MIT Press, Cambridge, Second printing, ISBN 0-262-01097-6, 1988.

J. Hertz et al. ont publié un ouvrage introductif, présentant une approche plutôt théorique des réseaux neuronaux (nombreuses équations), très clair et complet :

J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation, A Lecture Notes Volume in the Santa-Fe Institute Studies in the Sciences of Complexity*. Redwood City: Addison-Wesley Publishing Company, ISBN 0-201-51560-1, 1991.

L'approche employée par J. Dayhoff est plus biologique, et très pédagogique :

J. Dayhoff *Neural Networks Architectures* ; Van Nostrand Reinhold, ISBN 0-442-20744-1, 1990.

Neural Networks : le journal de INNS (cf. association) est "la" revue scientifique du domaine (Pergamon Press Inc. 395 Saw Mill River Road, Elmsford, New York 10523, USA).

IEEE Neural Networks : journal de l'IEEE, présente le même intérêt que le précédent (445 Hoes Lane, P. O. Box 1331, Piscataway, NJ 08855-1331, USA)

Il existe de nombreux autres journaux consacrés directement au connexionnisme, ou à ses domaines d'applications privilégiés.

Contacts en France et à l'étranger (liste non exhaustive)

News group : comp.ai.neural-nets
INNS-L@UMDD.bitnet (dépend de l'INNS)

Associations :
IEEE Neural Networks Council, cf. son journal.
INNS International Neural Network Society, Suite 300, 1250 24th Street, NW, Washington, DC 20037, USA. (Souscrire à l'association permet de s'abonner en même temps à la revue pour un prix modique).
NSI : Association Neurosciences et Sciences de l'Ingénieur, LTIRF, 46 avenue F. Viallet, Grenoble. (Organise chaque année, soit les journées NSI, soit une école de printemps).
AFCET, Groupe de travail "Information et Systèmes", 156 bd. Périere, 15017 Paris.
Club CRIN "Réseaux de neurones", Association ECRIN, 28, rue Sainte Dominique, 75007 Paris. (Destinée à favoriser les échanges Recherche-Industrie).
Greco CNRS (Groupes de Recherches Coordonnées), Projet Commande Symbolique et Neuromimétique.
ACTH : Association des Connexionnistes en THèse, LERI, Parc G. Besse, 30000 Nîmes.
ARC : Association pour la Recherche Cognitive, ENST, 46 rue Barrault, 75634 Paris Cedex 13.

Enseignements :
Ecole Polytechnique Fédérale de Lausanne (EPFL), cours Postgrade en Informatique Technique, réseaux de neurones biologiques et artificiels (180h), Dépt. informatique, IN-Ecublens, 1015 Lausanne, Suisse.
Ecole pour les Etudes et la Recherche en Informatique et Electronique (EERIE), cours de spécialisation 3ème année d'ingénieur (60h depuis 1990), Parc G. Besse, 30000 Nîmes, France.
Institut National Polytechnique de Grenoble, Sup-Telecom (Paris), Ecole des Mines d'Alès, Ecole Supérieure de Physique Chimie Industrielle (Paris), etc, les enseignements de DEA Neurosciences, Sciences Cognitives, etc.

Manisfestation :
Neuro-Nîmes : les réseaux de neurones et leurs applications. Chaque année à Nîmes depuis 1988, en novembre (conférence scientifique, cours, exposition de matériels). Contact : EC2, 269-287, rue de la Garenne, 92000 Nanterre, France.

Entreprise :

Neurosystèmes, Parc Scientifique Georges Besse, 30000 Nîmes.

La plupart des grandes entreprises (Thomson-CSF, Philips, Siemens, Alcatel, CNET, CEA-LETI, etc.) ont aussi leur propres services de recherches et développements.

18 Petit glossaire

Les définitions fournies ici ont pour objectif d'éclairer la lecture de cet ouvrage. Leurs validités est, a priori, restreintes.

Afférent :	Entrée.
Apprentissage :	L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.
Auto-association :	La clef est identique à la réponse.
Base d'apprentissage :	Exemples utilisés pour l'apprentissage, représentatifs du comportement de la fonction à réaliser.
Base de test :	Exemples différents de ceux de la base d'apprentissage, utilisés pour mesurer les performances de la généralisation.
Champ récepteur :	Région de l'espace d'entrée en relation avec un neurone (particulièrement en vision).
Clusterisation :	Terme dérivé de l'anglais identique à coalescence, qui lui est français.
Coalescence :	Regoupement des données.
Compétition :	Entre les neurones par des connexions latérales inhibitrices (exemple : ART, carte auto-organisatrice).
Cognition :	Ensemble des sciences dont l'objectif est la compréhension du conscient (Neurosciences, Informatique, Psychologie, etc.).
Connexionnisme :	Discipline définie par l'utilisation des réseaux de neurones artificiels pour l'ingénieur.
Dendrite :	Une!
Efférent :	Sortie.
Emergence :	Il y a émergence lorsque aapparait une propriété nouvelle non directement prédictible depuis la composition des éléments constitutifs ("la somme est supérieure à l'ensemble des parties").
Ethologie computationelle :	Utilisation de l'outil informatique pour l'étude du comportement animal (en particulier pour réaliser des simulations).
Généralisation :	Principal intérêt des réseaux de neurones artificiels. Capacité à répondre correctement à des situations inconnues (c.a.d. n'apparaissant pas dans la base d'apprentissage).
Hétéro-association :	La clef est différente de la réponse.
Informatique :	Science du traitement automatique de l'information, plus ancienne que les ordinateurs.

Mémoire associative :	Mémoire adressable par son contenu, à la différence des mémoires plus classiques ou l'adresse est requise.
Neuromimétique :	Discipline définie par l'utilisation des réseaux de neurones artificiels pour valider des hypothèses biologiques par le neuroscientiste (!).
Neurone :	Unité de traitement de l'information dans le cerveau, au nombre mille milliards environ.
Neurosciences :	Ensemble des sciences ayant pour objectif l'étude du cerveau, depuis le niveau moléculaire jusqu'aux comportements sociaux en passant par les réseaux de neurones (Histologie, Neuropsychopharmacologie, Neurophysiologie, Ethologie, Psychologie, etc.).
Organisation :	Ce n'est pas mettre de l'ordre! C'est , au sens biologique, générer de la vie.
Pattern matching :	Terme anglais identifiant l'opération de mise en correspondance de deux formes.
Processeur élémentaire :	Allusion au processeur de calcul (en beaucoup plus simple), aussi appelé neurone ou cellule.
Réseaux de neurones artificiels :	Réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.
Supervisé :	Les poids sont modifiés en fonction de la sortie désirée.
Non supervisé :	Les sorties désirées sont inconnues, les entrées sont projetées sur l'espace du réseau.
Synapse :	Jonction entre deux neurones, très complexe au niveau des mécanismes chimiques mis en oeuvre ; outrageusement simplifiée dans les modélisations.
Vie artificielle :	L'objectif de cette nouvelle (!) discipline est de créér des êtres artificiels. Le connexionnisme est une des techniques impliquées.